VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

MULTIPLATFORMNÍ MOBILNÍ APLIKACE NAHRAZUJÍCÍ VČELAŘSKÝ DENÍK S PROPOJENÍM NA ÚLOVÉ VÁHY

MULTIPLATFORM MOBILE APP APIARY BOOK FOR BEEKEEPERS WITH CONNECTION TO HIVE SCALE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **PETR DRÁBEK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. JAN PLUSKAL**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Drábek Petr**
Program: Informační technologie
Název: **Multiplatformní mobilní včelařský deník**
Multiplatform Mobile Beekeeping Diary
Kategorie: Vestavěné systémy

Zadání:

1. Proveďte analýzu včelařského deníku a vytvořte specifikaci požadavků na včelařský deník, kterou podpoříte zpětnou vazbou získanou od potenciálních uživatelů.
2. Navrhněte mobilní aplikaci nahrazující úlový deník pro usnadnění práce včelařům s možností propojení na úlovou váhu. Proveďte návrh úlové váhy za pomoci volně dostupných IoT komponent.
3. Řešení implementujte jako mobilní aplikaci pro prostředí Xamarin a vytvořte funkční vzorek úlové váhy. Úlová váha a mobilní aplikace bude využívat cloudových PasS služeb.
4. Otestujte mobilní aplikaci, nabídněte ji potenciálním uživatelům a získejte zpětnou vazbu. Výsledky diskutujte.

Literatura:

- Hermes, D. (2015). Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. Apress, Berkeley, CA.
- Riti, P. (2018). Introduction to GCP. In *Pro DevOps with Google Cloud Platform* (pp. 19-35). Apress, Berkeley, CA.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Pluskal Jan, Ing.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 25. října 2019

Abstrakt

Cílem práce je vytvořit přehlednou a intuitivní mobilní aplikaci nahrazující klasické papírové zaznamenávání údajů o včelstvech pro včelaře. Poskytnout možnost propojení této aplikace s úlovou váhou pro ještě větší přehled o stavu včelstev. Aplikace je vyvíjena na platformě Xamarin pro operační systémy Android a iOS. Výsledek této práce má sloužit včelařům k usnadnění práce se včelami, ukládání jejich dat do databáze a umožnit jim s nimi pracovat na různých zařízeních. Poskytnout včelaři aktuální přehled o jeho stanovištích, včelstvech a úlech, ale také přehledné statistiky z těchto dat.

Abstract

The aim of this work is to create a transparent and intuitive mobile application replacing the classic paper recording of data about bee colonies for beekeepers. Furthermore, to provide the possibility of linking this application to the hive scale for even greater insight into bee colonies. The application is developed on Xamarin platform for Android and iOS. The result of this work is to help beekeepers to work with bees, store their data in a database and enable them to work with them on different devices. Provide the beekeeper with an up-to-date overview of their apiaries, colonies and hives, as well as statistics from this data.

Klíčová slova

včelaření, včelařský deník, mobilní aplikace, xamarin, multiplatformní programování, úlová váha, IoT

Keywords

beekeeping, beekepers diary, mobile application, xamarin, multiplatform programing, bee hive scale, IoT

Citace

DRÁBEK, Petr. *Multiplatformní mobilní aplikace nahrazující včelařský deník s propojením na úlové váhy.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Pluskal

Multiplatformní mobilní aplikace nahrazující včelařský deník s propojením na úlové váhy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jana Pluskala. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Petr Drábek

27. května 2020

Obsah

Kapitola 1

Úvod

Podle organizace Earthwatch Institut jsou včely nejdůležitější živá stvoření na Zemi, jak rozhodla na poslední schůzi Královské geografické společnosti v Londýně [3]. Na včelách závisí asi 70 procent světového zemědělství. Opylováním zajišťují rozmnožování řadě rostlin, které prý odhadem poskytují potravu až 90 procentům lidské populace [11]. Lidé si čím dál více uvědomují, jak je včela pro lidstvo důležitá.

Včelaření se za poslední roky stává čím dál větším trendem hlavně mezi střední generací, která je pracovně hodně vytížená a nevěnuje včelám tolik pozornosti, kolik by si zasloužily [7]. Tato generace je jedna z cílových, pro kterou je výsledek této práce určen. Přehled o svých včelstvech budou mít neustále u sebe ve svém mobilním telefonu. Budou moci rychleji rozpoznat slabé a nemocné včelstvo, které může být zdrojem šíření nákazy kleštíka včelího i pro okolní včelaře. Propojení včelaření s moderními technologiemi v nich doufám také prohloubí zájem o včelaření, studium literatury a zlepší úroveň stavu jejich včelstev.

Další skupinou, na kterou výsledek práce cílí, jsou zkušenější včelaři, kteří mají včelaření jako koníček. Ví o svých včelách hodně a starají se o ně pečlivě. Dá se říct, že je to jejich životní záliba. Takový včelař si bude s radostí zaznamenávat podrobně všechna data a sledovat vývoj váhy včelstva.

Poslední cílovou skupinou jsou komerční včelaři. Zřejmě nebudou chtít úlové váhy masivně pro svá včelstva, ale určitě ocení přehlednost a rychlost zadávání dat o svých včelstvech. Data budou moci navíc sdílet napříč několika zařízeními, která mohou využívat jednotliví zaměstnanci. Pokud tedy bude včelstvo ošetřovat jiný člověk než doposud, lehce zjistí z aplikace předchozí stav včelstva.

Cílem práce je vytvořit komplexní včelařský deník v podobě mobilní aplikace, která umožní včelařům jednoduše evidovat všechna data o jejich činnostech a včelstvech jednoduše na mobilním zařízení. Včelař si bude moci evidovat svá stanoviště, včelstva a zaznamenávat činnosti jako jsou prohlídky, medobraní, krmení a léčení. Všechna zaznamenaná data budou včelaři přehledně přístupná v aplikaci, a to i z více zařízení současně. Aplikace bude také použitelná i bez přístupu k internetu, aby si včelař mohl zaznamenávat data přímo na stanovišti, kde internet nemusí mít k dispozici. Součástí práce je také vytvoření prototypu úlové váhy, která bude s aplikací komunikovat. Cílem je demonstrovat, jaké výhody toto propojení může včelaři přinést, není však cílem nahradit kompletní řešení chytrých úlů, taková řešení již existují a jsou dobře propracovaná. Hlavním cílem je uživatelsky přívětivý, komplexní včelařský deník v podobě aplikace, která na trhu tolik chybí.

Kapitola 2

Včelařský deník

Včelařský deník slouží pro zaznamenávání včelařských aktivit a informací o včelstvech. Pro různé včelaře může mít různé podoby. Nejčastější řešení je zatím zaznamenávání informací do klasického papírového deníku, nevýhodou je zdlouhavé zapisování, kdy si musíte všechno zapsat ručně a navíc můžete takový deník lehce ztratit a přijít o všechny informace. Další možností úlového deníku je zapisovat si informace přímo na úl nebo na fólii pod víkem, největší nevýhodou je malé množství informací, které tímto způsobem lze zapsat a možnost, že se záznamy mohou vlivem přírodních živlů smazat. Mezi další řešení patří zaznamenávání informací do počítače, většinou do excelových tabulek, takové řešení je asi časově nejnáročnější, ovšem výhodou je, že si uživatel může pomocí vzorečků vést statistiky ze zaznamenaných dat, jako součet vytočeného medu za sezónu, množství celkově dodaného krmiva a podobné. Poslední možností je použití aplikace přímo určené k zaznamenávání včelařských aktivit, většinou mobilní. Aplikace může ušetřit spoustu času ať už se zaznamenáváním samotných informací nebo poskytnutí přehledného souhrnu dat a statistik.

Vedení včelařského deníku má mnoho výhod. Nejdůležitější je přehled, který včelař získá o svých včelstvech a stanovištích. Může hodnotit historický vývoj. Jednoduše zjistí, v jakém stavu se nacházelo včelstvo při poslední prohlídce, může sledovat výkonnost jednotlivých matek a mnoho dalšího. Včelařský deník v konečném důsledku usnadní včelaři práci a pomůže mu se lépe starat o svá včelstva.

2.1 Úvod do včelařské terminologie

Aby byla práce čitelná i pro čtenáře nezasvěceného do včelařství, je nutné vysvětlit některé pojmy, které se budou v práci objevovat. Není možné a ani není účelem práce rozebrat všechny včelařské pojmy, které se zde objevují. Mezi základní terminologii patří:

Stanoviště Je místo, kde se pohromadě nachází jeden či více úlů. V České republice musí mít každé stanoviště přiděleno jednoznačné registrační číslo, kvůli pravidelnému a povinnému hlášení počtu včelstev.

Úl Je člověkem vytvořený příbytek pro chov včelstva. V každém úlu se nachází právě jedno včelstvo. V přírodě včelám k životu stačí tmavá a suchá dutina, do které vede jen malý otvor, aby si dokázaly uhájit domov před vetřelci, a ve kterém zrovna nikdo nebydlí [22]. V dnešní době tomu již tak není a včely medonosné se vyskytují převážně jen v uměle vytvořených úlech, které se skládají z úlového dna, nástavků a stříšky. V nástavcích se pak nachází rámky s včelím plodem, pylem a medem.

Včelstvo Včela medonosná vytváří početná společenstva – včelstva, která jsou tvořena včelími jedinci — včelí matkou (vždy jen jedna), dělnicemi a trubci, včelím dílem (tj. plásty s dělničími buňkami, trubčími buňkami a dočasnými matečníky), v kterém probíhá vývoj včelího plodu různého stáří (vajíčko, larva, kukla), a do kterého včelí dělnice ukládají medové a pylové zásoby, které slouží včelám jako potrava [18].

Včelí matka/královna Nejdůležitější úlohou včelí matky ve včelstvu je udržení včelího rodu. Matka je jedinou samičkou ve včelstvu, která má dokonale vyvinuté pohlavní orgány (párové vaječníky), díky čemuž se může na snubních letech pářit s trubci a klást i oplozená vajíčka, z kterých se líhnou další samičky — dělnice a matky [18].

Matka bývá značena barvou nebo nálepkou, pro usnadnění hledání matky ve včelstvu. Typicky se barva značky vybírá podle daného roku, ve kterém se matka narodila. Výběr barvy má přesně daný systém, aby bylo možné ihned odhadnout stáří matky podle zvolené barvy. Barva se vybírá podle poslední číslice daného roku, pro koncové číslice:

- **1 nebo 6** — barva bílá
- **2 nebo 7** — barva žlutá
- **3 nebo 8** — barva červená
- **4 nebo 9** — barva zelená
- **5 nebo 0** — barva modrá

Nástavek Nástavky jsou převážně dřevěné boxy, které slouží k vytvoření těla úlu. Skládají se na sebe, a tím je možné zvětšovat a zmenšovat úlový prostor. Jsou vyplněny rámky.

Rámek Rámky mohou být jak dřevěné (nejpoužívanější), tak plastové. Často je vydrátkován tenkým drátkem, sloužícím ke zvýšení pevnosti včelího díla. V průběhu cyklu životnosti rámku si nese v rámci včelařské terminologie různé přívlastky nebo názvy:

- **Stavební rámek** — prázdný rám, ještě před vložením do úlu. Slouží pro volnou stavbu včelího díla.
- **Mezistěna** — rámek vyplněn mezistěnou. Mezistěna je vylisovaná šablona pro stavbu včelího díla z včelího vosku. Určuje velikost buněk.
- **Souše** — již vystavěný stavební rámek nebo mezistěna. Je to včelí dílo tvořeno z vosku, u kterého se v jednotlivých buňkách zrovna nic nenachází.
- **Plodový rámek** — vystavěný rámek s včelím plodem. Často se na něm vyskytuje i pyl a med.
- **Pylový a medový rámek** — vystavěný rámek, ve kterém se nachází pouze pyl nebo med.

Medobraní Je událost, kdy včelař odebírá med ze svých včelstev. Nejprve vybere od včelstev rámky plné zralého medu a ty poté ve stáčecí nádobě vytáčí, filtruje a případně vyfiltrovaný med plní do sklenic.

Krmení Při medobraní včelař odebere od včelstev zásoby medu, které si v průběhu jara a léta včely nasbíraly na zimu. Tyto zásoby jim musí včelař zpátky doplnit, děje se tak převážně cukrem. Je důležité, aby měl včelař přehled, kolik a v jakém poměru přidal

krmení jednotlivým včelstvům. Může se totiž stát, že včelstvo překrmí a matka poté nebude mít prostor pro kladení nových vajíček nebo naopak včelstvo přikrmí málo, a to pak v průběhu zimy uhyne hlady.

Léčení Při léčení se ve včelařském světě snaží včelař zmírnit výskyt kleštíka včelího ve včelstvu. Kleštík včelí parazituje na plodu i dospělých včelách a je šiřitelem virových onemocnění, které dokážou ve větším množství zlikvidovat včelstvo. Kleštík včelí je hlavní příčinou masivního úhynu včelstev v posledních letech. Včelař musí mít přehled, kdy a jak prováděl léčení, na kterých včelstvech, a také by měl monitorovat spad kleštíka včelího po jednotlivých léčení, aby měl průběžný přehled o jeho výskytu ve včelstvu a mohl proti němu provádět patřičná opatření.

Prohlídka Jednoduše řečeno jedná se o kontrolu stavu včelstva. Při prohlídce včelař kontroluje sílu včelstva, stav zásob, množství plodu, přítomnost vajíček, matky a další ukazatele dobrého stavu včelstva. Podle prohlídky plánuje další opatření, která bude se včelstvem provádět. Pokud například při prohlídce včelař zjistí, že ve včelstvu chybí matka, obstará do následující prohlídky včelstvu matku novou nebo vloží rámek s otevřeným plodem do takového včelstva a nechá včelstvu vychovat si matku vlastní.

2.2 Analýza existujících řešení

V současné době existuje několik řešení úlového deníku v podobě mobilní aplikace. Téměř všechny tyto aplikace jsem si vyzkoušel, každá nabízí jiné funkce, ale všem jde o jediné, umožnit uživateli zaznamenávat si data týkající se včelaření. Většina z nich se netěší oblibě hlavně kvůli chybějícím funkcím, nepřehlednosti, a hlavně ztrátě podpory.

2.2.1 Srovnání existujících řešení v podobě mobilní aplikace

Jako součást průzkumu trhu jsem se blíže zaměřil na aplikace zabývající se problémem zaznamenávání včelařských aktivit. Vybral jsem několik klíčových vlastností a sestavil tabulku 2.1 s jednotlivými aplikacemi a jejich vlastnostmi.

Aplikace	Data v cloudu	Offline sync.	Uživ. prostředí	Ovládání	Chyby
ApiManager	ano/předplatné	ne	přívětivé	zbytečně složité	ano
Deník včelaře	ano, pouze předplatné	ne	postarší	intuitivní	ano
Včelař	ne	ne	jednoduché	jednoduché	ne
Hive Manager	ano/předplatné	ne	jednoduché	intuitivní	ano
HiveKeepers	ano	ne	přívětivé	složité	ano

Tabulka 2.1: Srovnání mobilních aplikací nahrazujících včelařský deník.

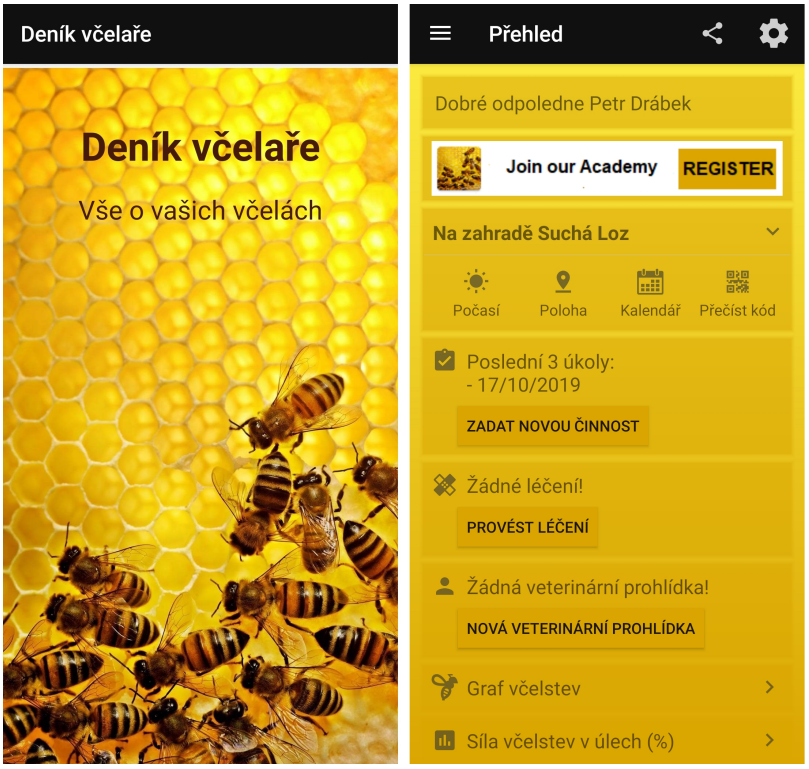
Do tabulky jsem dodatečně přidal také sloupec s vlastností, zdali se mi podařilo v aplikaci najít zřetelné chyby. To se mi bohužel podařilo u většiny. Převážně se jednalo o chyby ve spojitosti s odpojením a připojením od internetu v průběhu používání. Pozorované aplikace používají buď lokální nebo serverovou databázi. Aplikace s databází na serveru nebylo možné používat v režimu bez připojení k internetu. Obecně můžeme říci, že se s těmito aplikacemi táhne množství problému. To byl také důvod, proč jsem se rozhodl vytvořit pro včelaře vyhovující aplikace v rámci bakalářské práce. Z pozorování jsem také čerpal inspiraci pro mou práci, abych poté mohl uživateli poskytnou všechny klíčové vlastnosti v

plně funkční aplikaci. V následující sekci se blíže podíváme na nejpoužívanější z testovaných aplikací.

2.2.2 Mobilní aplikace Deník včelaře od Bogdana Iordache

Jak bylo již řečeno, tato aplikace je, co se týče oblíbenosti uživatelů, nejúspěšnější. Má podporu, dlouhou historii a širokou základnu uživatelů. Nese přímo název Deník včelaře[1] a je od maďarského vývojáře Bogdana Iordache.

Mezi hlavní výhody této aplikace patří dostupnost v celkově 17 jazycích, ke kterým patří i čeština, dále dostatek funkcí, stabilní podpora a možnost zálohovat si data v cloudu. Uživatel si může zaznamenávat stanoviště, úly, matky, výnosy, léčení, veterinární prohlídky, poznámky a inventář.



Obrázek 2.1: Deník včelaře od Bogdana Iordache.

Aplikaci jsem aktivně testoval a používal. Jako nedostatky bych určitě uvedl nepřehlednost, opravdu trvá, než se zorientujete a zjistíte, co musíte vykonat pro zaznamenání si konkrétní činnosti. Například pokud chcete zaznamenat přidání mezistěn do úlu, musíte provést následující kroky:

1. Rozkliknout menu přehled.
2. Z menu vybrat položku úly.
3. Zvolit konkrétní úl.
4. Přejít do sekce úlové části.

[1] https://play.google.com/store/apps/details?id=com.csg.apiarybook&hl=cs

5. Kliknout na tlačítko přidat.
6. Vybrat položku mezistěny/souše a zadat počet.
7. Následně uložit.

Přitom by mohl jít jednoduše zadat počet přidaných mezistěn při prohlídce. Aplikace navíc nerozlišuje přidání mezistěny a souše, mezi kterými je podstatný rozdíl. Drobný nedostatek, ale nepříjemný jsem také shledal v tom, že pokud přidáte jakoukoliv položku, zobrazí se pouze zpráva o úspěchu, ale již se nepřejde na předchozí stránku, a tak musíte sami zadat tlačítko zpět. Naopak velký nedostatek vidím v tom, že aplikace sama nezálohuje data na cloud, ale pro jejich zálohu v cloudu musíte přejít do nastavení, zálohy a obnovy a dvakrát kliknout na zálohovat. Jak zálohování funguje spolehlivě, jsem již nezkoumal. V aplikaci se dá využít dostatek funkcí, i když jsou některé pro včelaření zbytečné. Jedná se například o vedení si adresáře kontaktů v deníku nebo sdílení přátelům, že jste na stanovišti. Některé funkce jsou naopak nefunkční, jako přidání fotografie u stanoviště nebo vybrání pozice stanoviště na mapě.

Pro nenáročného uživatele je však aplikace i přes zastaralejší vzhled určitě použitelná. Všechny činnosti si v ní včelař může evidovat a zpětně zobrazovat. Ovšem kvůli složitějšímu používání a jejím nedostatkům už si nejsem jistý, jak dostatečně dokáže včelaři šetřit čas a ulehčit práci. V aplikaci mi také chybí jednoduchý přehled a statistiky z dat, která uživatel zadává.

2.3 Průzkum mezi uživateli

Problémem evidování včelařských aktivit se zabývám již několik let. Před třemi lety jsem vytvořil již jeden úlový deník pro telefony s Androidem, ovšem s omezenou funkčností, pouze lokální databází na zařízení a bez profilu uživatele. I tak se aplikace dočkala obliby, nyní má něco přes tři stovky aktivních uživatelů, i přesto že je pouze v českém jazyce, dokonce o ní vyšel článek[2]. Za celou dobu, co je aplikace vydaná pro veřejnost, mi od uživatelů přicházela zpětná vazba. Konzultoval jsem s nimi zkušenosti a návrhy na zlepšení. Mým cílem je, aby se všechny postřehy, co jsem za ty roky nasbíral projevily ve výsledku této práce.

Je pro mě důležité, aby aplikace byla přesně podle představ včelařů, má jim přeci dobře sloužit. I proto jsem oslovil širší okruh včelařů, a to za pomocí dotazníku. První část dotazníku je zaměřena na to, jestli uživatelé používají nějaký včelařský deník a případně jaký. Uživatelů jsem se také ptal, pokud používají papírový úlový deník, zdali by byli ochotni přejít na mobilní nebo desktopovou aplikaci. V další části dotazníku jsem vznesl dotaz, jaká konkrétní data shledávají užitečnými pro zaznamenávání do úlového deníku. Podařilo se mi oslovit celkem 162 českých a slovenských včelařů. U anglicky mluvících kolegů jsem již tak úspěšný nebyl a podařilo se mi získat informace pouze od 26 včelařů. I proto jejich odpovědi nebudu v práci uvádět, nicméně v návrhu aplikace je zohledním.

[2] https://androidaplikace.cz/index.php/2018/05/i-vcelarstvi-jde-s-dobou-a-modernizuje-se-vcelarum-skvele-poslouzi-aplikace-ulovy-denik-vcelare/?fbclid=IwAR38s3dYc6aWFZq-NXWY3BPJCzS9pTJ4aoBGOe9ZgkMyr5AQlVHVCJn4lRg

2.4 Zhodnocení dotazníku

Na otázku 2.2, zda respondent používá úlový deník, případně jakou formu, odpovědělo 54,3 %, že používají papírovou formu úlového deníku. U těchto respondentů jsem se dále tázal, zdali je u nich možnost, že by přešli na některou z mobilních nebo desktopových aplikací, pokud by jim vyhovovala. 79,5 % z nich odpovědělo ano. Z toho vyplývá, že je ještě velké množství potencionálních uživatelů mezi včelaři, kteří by mohli mou aplikaci využívat.

Obrázek 2.2: Jaké používají respondenti úlové deníky.

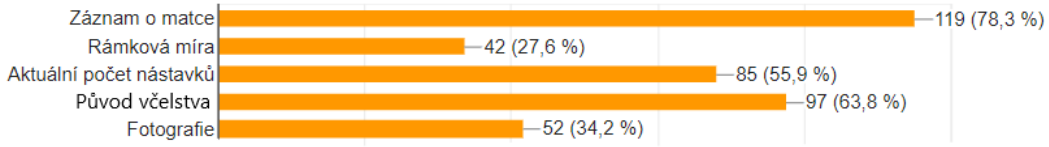
Jak můžete vidět výše, druhou nejpočetnější skupinou, celkem 24,7 %, jsou včelaři, kteří si údaje o své včelařské činnosti nijak nezaznamenávají. I na tuto skupinu bych chtěl práci mířit a motivovat je propracovanou aplikací k vedení si evidence, která by jim umožnila lépe se starat o svá včelstva.

Pouze 9,9 % respondentů již používá některou z dostupných aplikací. To je podle mého názoru hodně málo a vidím zde velkou příležitost ke zlepšení. Zbylých 11,1 % respondentů odpovědělo individuálně. Mezi nejčastější individuální odpovědi patří používání tabulkového procesoru Microsoft Excel, dělání si popisků přímo na úl, používání google docs nebo že uživatelé teprve vhodné řešení hledají. I mezi touto skupinou se dá najít mnoho potencionálních uživatelů.

V další části dotazníků jsem se již ptal na konkrétní data, která si včelaři zaznamenávají k jednotlivým činnostem, stanovišti nebo úlu. Hned u první otázky 2.3 u údajů ke stanovišti mě překvapil největší zájem k zobrazování počasí na jednotlivých stanovištích. Tuto funkcionalitu tedy určitě do aplikace zavedu. Celkově zájem o údaje ke stanovištím měla menší polovina respondentů, nicméně data ke stanovišti zadá uživatel pouze jednou na začátku, nebudou tedy uživatele, kteří o ně nemají zájem nijak obtěžovat.

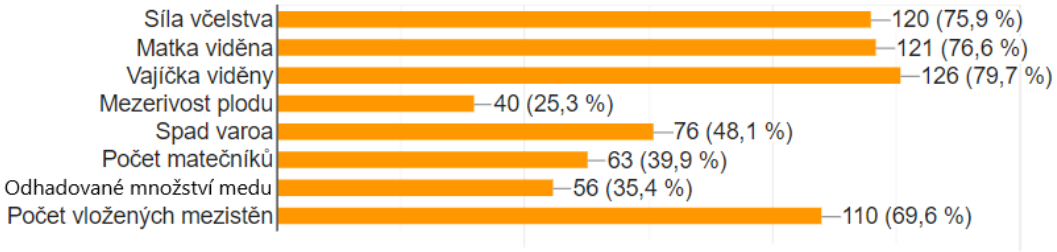
Obrázek 2.3: Informace, které by respondenti zaznamenávali u stanoviště.

U úlu 2.4 mají respondenti největší zájem o evidenci záznamu o matce. I proto jsem rozhodl, že se v aplikaci bude matka evidovat zvlášť, bude náležet ke konkrétnímu úlu a bude mít více vlastností. Naopak nejmenší zájem je o zaznamenávání rámkové míry a fotografií, pro zjednodušení nezakomponuji tyto položky do aplikace.



Obrázek 2.4: Informace, které by respondenti zaznamenávali u úlu.

Dle mých očekávání je u prohlídky 2.5 největší zájem o zaznamenávání síly včelstva, zda byla viděna matka a vajíčka. Naopak překvapením pro mě byl nezájem o zaznamenávání si mezerovitosti plodu a na druhou stranu veliký zájem o zaznamenání si počtu vložených mezistěn. Ve volných odpovědích se nejčastěji vyskytla potřeba zapsat si vykonané činnosti u dané prohlídky nebo poznámky.



Obrázek 2.5: Informace, které by respondenti zaznamenávali u prohlídky.

Největší zájem u respondentů vzbudilo zaznamenávání si léčení 2.6, téměř o všechny položky je více jak 50% zájem. Mezi volnými odpověďmi se několikrát objevilo od slovenských kolegů, že si musí vést záznamovou knihu o léčení. Kdyby si mohli zaznamenávat data rovnou do aplikace na včelnici, ulehčila by se jim práce.

Téměř u všech položek k evidenci se ve volných odpovědích vyskytly potřeby k zaznamenání si poznámky, tuhle skutečnost určitě vezmu v potaz. Celkově dotazník hodnotím velice prospěšně, hlavně co se týče toho, jaká data umožnit uživateli evidovat k jednotlivým položkám. V některých mě utvrdil a v jiných zase vyvedl z omylu. Na některé činnosti, jako jsou medobraní a krmení, jsem se respondentů neptal, protože není velká možnost, co si u

Obrázek 2.6: Informace, které by respondenti zaznamenávali při léčení.

těchto činností zaznamenávat. U medobraní to je datum, množství, druh medu, množství vody a poznámka. U krmení opět datum, množství, druh, způsob podání a poznámka. Všem respondentům bych chtěl poděkovat za jejich čas, kterým dopomohli vytvořit aplikaci na míru dělanou právě jim, včelařům.

Kapitola 3

Návrh databáze, včelařského deníku a prototypu úlové váhy

V následující kapitole se podíváme na datový návrh mobilní aplikace, rozdělení jednotlivých položek, které vychází z předchozích zkušeností a průzkumu mezi uživateli 2.3, do databázových tabulek, a nakonec výsledné schéma databáze. Dále na předběžný návrh uživatelského rozhraní aplikace a nakonec na schéma a výběr komponent úlové váhy.

3.1 Datový návrh mobilní aplikace

Data, která si budou moci včelaři v aplikaci zaznamenávat, vychází z průzkumu současných řešení, vlastních zkušeností, a hlavně z průzkumu mezi uživateli. Rozebereme si jednotlivé datové objekty a jejich atributy, k čemu vlastně mohou sloužit.

Stanoviště U stanoviště si uživatel bude moci zaznamenat název, popis, adresu, katastrální číslo pozemku, registrační číslo stanoviště a fotografii. Přičemž katastrální číslo pozemku a registrační číslo stanoviště slouží pouze včelaři, který tyto údaje musí odesílat v každoročním hlášení počtu včelstev.

Úl Úl bude vždy náležet ke konkrétnímu stanovišti. Zároveň reprezentuje i včelstvo, které v něm sídlí. Uživatel si u úlu může zadat počet nástavků, rámků, původ včelstva a název. Původ včelstva může být například vlastní oddělek, smetenec nebo zakoupené včelstvo a další.

Matka Matka vždy patří ke konkrétnímu včelstvu, v našem případě tedy k úlu. Nese atributy jméno, původ, označení, plemeno a přibližný datum vylíhnutí. Původ matky může být například z vlastního chovu, nouzového nebo rojového matečníku. Může a taky nemusí nést označení. Pokud matka označení má, je označena většinou barvou nebo nálepkou s číslem. Barva matky by měla také označovat rok jejího vylíhnutí, avšak ne všichni včelaři toto nepsané pravidlo dodržují. Datum vylíhnutí je důležitý k určení stáří matky, které může být důvodem k její výměně.

Prohlídka Prohlídka je přiřazována k úlu. Slouží uživateli k zaznamenání si současného stavu včelstva. Při následující prohlídce se může tedy včelař jednoduše podívat na předchozí prohlídku a podle stavu včelstva už dopředu vyhodnotit, jaké úkony bude ve včelstvu vykonávat. U prohlídky si uživatel bude moct zaznamenat datum, sílu včelstva, zda viděl matku a vajíčka, spad varroa destructor, matečníky, počet rámků

s medem, plodem, pylem, jestli také přidal nástavek, mezistěny, souše nebo stavební rámky a poznámky.

Léčení Kvůli napadení včelstev parazitem *varroa destructor* jsou všichni včelaři povinni svým včelstvům pomáhat zbavit se tohoto parazita za pomocí dostupných léčiv. *Varroa destructor*, česky kleštík včelí, se k nám dostal pravděpodobně na začátku 20. století po dokončení transsibiřské magistrály z Asie. Rychle se v Evropě rozšířil a jelikož není evropská včela medonosná geneticky ani jinak vybavená k potlačení rozvoje tohoto cizopasníka, stojí za naprostou většinou úhynu včelstev a kolapsem celých kolonií.

Léčení je tedy opravdu důležité a pro přehled, kdy a jak byla jednotlivá včelstva léčena, je dobré si jej zaznamenávat. Uživatel si může v aplikaci u léčení evidovat datum, poznámku, druh léčiva, množství léčiva, způsob podání a dodatečný spad *varroa destructor*.

Medobraní a krmení Poslední položky, co se týče včelařské činnosti, jsou medobraní a krmení. Medobraní je činnost, kdy včelař získává ze svých včelstev med. Může si u něj zaznamenat datum, hmotnost, množství vody a libovolnou poznámku. Naopak při krmení včelař včelstvu potravu dodává. Ke krmení si může zaznamenat datum, druh krmiva, množství, způsob podání a poznámku.

Úlová váha a záznamy z vážení V aplikaci se budou ukládat samozřejmě také záznamy z úlových vah. Záznam z vážení bude obsahovat datum, hmotnost a bude vždy náležet ke konkrétnímu úlu.

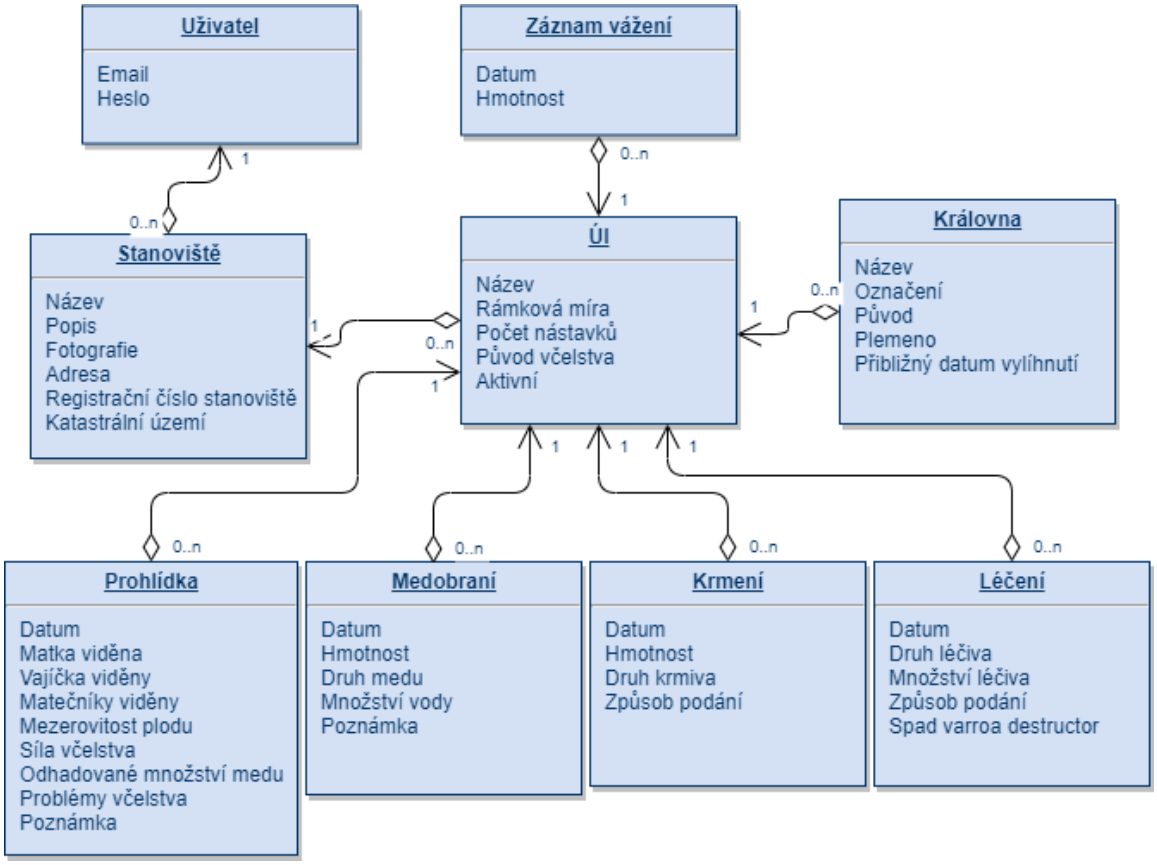
3.2 Entitně-vztahový model

V informačních systémech se o entitně-vztahovém, jinak také ER modelu, bavíme jako o modelu, který představuje objekty reálného světa a vztahy mezi nimi. Slouží nám k návrhu databáze.

Základní pojmy

- **Entita** — je objekt reálného světa, rozlišitelný od jiných objektů. (V našem případě například **Úl** s názvem `Úl01`.)
- **Atribut** — představuje vlastnost entity. (Název úlu, počet nástavků, původ včelstva a další.)
- **Vztah** — vyjadřuje spojitost mezi entitami. (Úl s názvem `Ul01` **náleží** ke stanovišti s názvem `Na zahradě`.)
- **Entitní množina** — množina entit stejného typu se stejnými vlastnostmi. (Úl, stanoviště.)
- **Vztahová množina** — množina vztahů stejného typu se stejnými vlastnostmi. (Úl **náleží** ke stanovišti.)

Na obrázku 3.1 lze vidět návrh v podobě entitně-vztahového modelu pro databázi.



Obrázek 3.1: Zjednodušený entitně-vztahový model mobilní aplikace.

Kardinalita

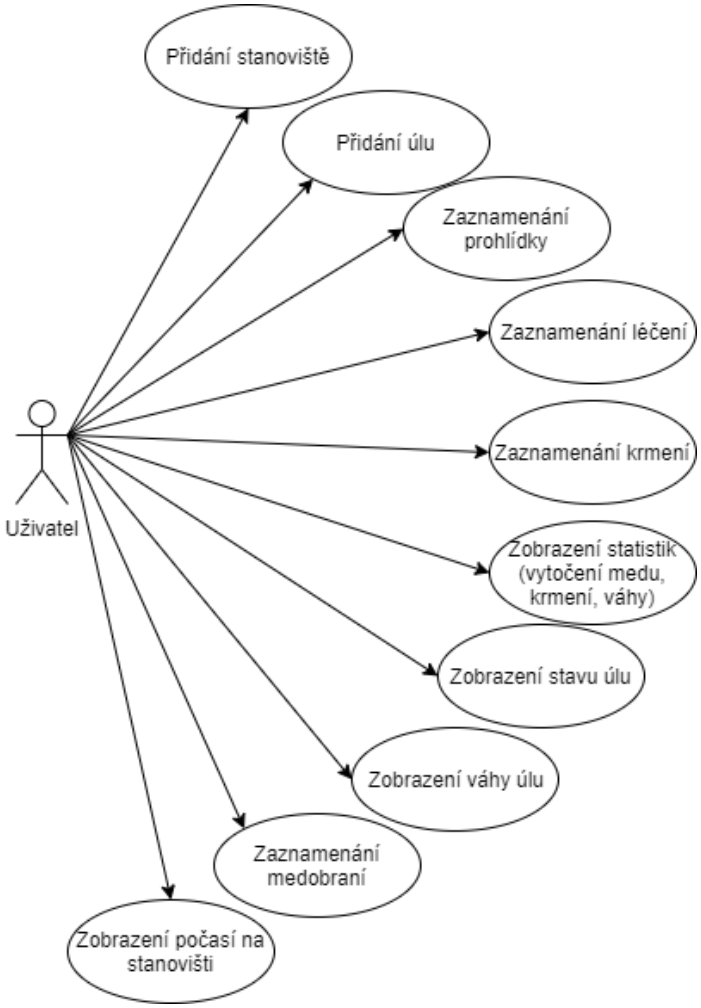
Popisuje, kolikrát se každá instance dané entity může účastnit daného typu vazby. Jinak řečeno, kolik výskytů jedné entity může vstoupit do vztahu s kolika výskyty druhé entity. Například k jednomu konkrétnímu stanovišti může náležet nula až několik konkrétních úlů. Rozlišujeme několik druhů vazeb:

- **1:1** - jedna instance první entity je ve vazbě právě s jednou instancí druhé entity. (Manžel má manželku a manželka má manžela.)
- **1:N** - jedna instance první entity může vstoupit do vztahu s libovolným počtem instancí druhé entity. (Jedno stanoviště může obsahovat několik úlů.) Tuto vazbu budeme využívat v našem modelu s označením `1:0..n`, aby bylo jasnější, že jedna instance první entity může vstoupit do vztahu s nula až libovolným počtem instancí druhé entity.
- **M:N** - k jedné instanci první entity může vstoupit do vztahu libovolný počet instancí druhé entity a naopak k jedné instanci druhé entity může vstoupit do vztahu libovolný počet instancí první entity. (Jedna řeka může protékat více státy a jedním státem může protékat více řek.)

3.3 Diagram případů užití

Diagram případů užití 3.2 slouží k zachycení vnějšího pohledu na systém, a pomáhá odhalit hranice systému. Slouží k definování chování, jakým způsobem chce uživatel aplikaci používat. Někdy je mylně zamýšleno, že diagram případů užití slouží k zachycení všech úkonů, které uživatel s aplikací bude provádět. Ale kroky, které nepopisují účel systému jako přihlášení, či registrace, do diagramu případů užití nepatří.

Diagram by měl modelovat funkčnost systému za pomocí herců a případů užití. Případy užití jsou sady akcí, služeb a funkcí, které má systém provádět [10]. V našem případě je systém mobilní aplikace a herec včelař, který aplikaci používá.



Obrázek 3.2: Diagram případů užití mobilní aplikace Deník včelaře.

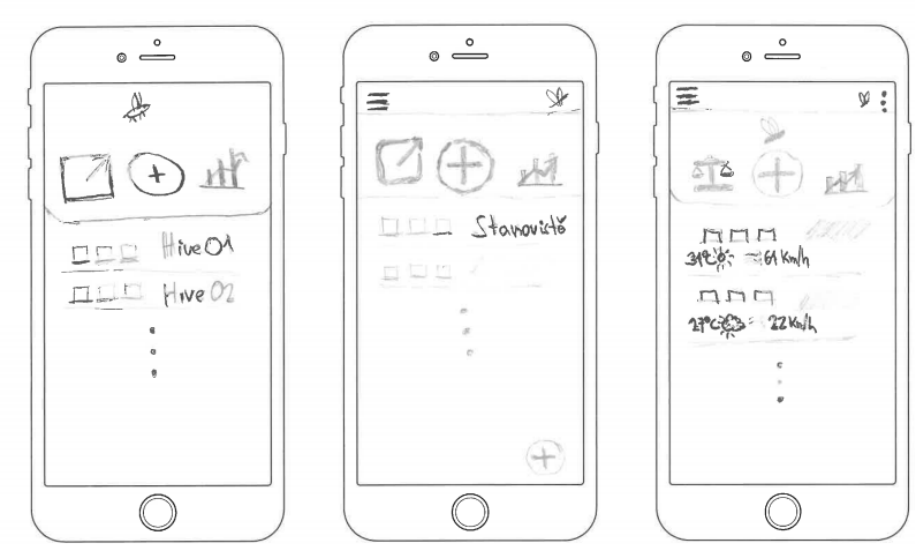
3.4 Návrh GUI mobilní aplikace

Po úspěšném získání dat od uživatelů a sestavení datového modelu můžeme přejít na návrh samotného grafického uživatelského rozhraní, tzn. jak bude aplikace vypadat a jak ji bude možné ovládat.

3.4.1 Tvorba skic

Ruční skicování (angl. sketch) je rychlý a jednoduchý nástroj k převodu myšlenek na papír. Jde o velice rychlý a snadný způsob, jak lze vizualizovat první návrh samotné aplikace. Je založen na principu, který říká, že i jednoduchý náčrt může vyjádřit a popsat nápad mnohem lépe než slova. Při vytváření skic vznikají nové nápady na vylepšení aplikace. Umožňuje vytvořit základ vzhledu a rozložení jednotlivých obrazovek. K vytvoření skic jsem použil klasickou tužku a vytištěné šablony pro skicování mobilních aplikací se vzorem telefonu iPhone 6.

U některých obrazovek je dobré si udělat skic více a vybrat z nich až následně tu nejlepší. Tento způsob jsem zvolil u návrhu hlavní obrazovky 3.3.



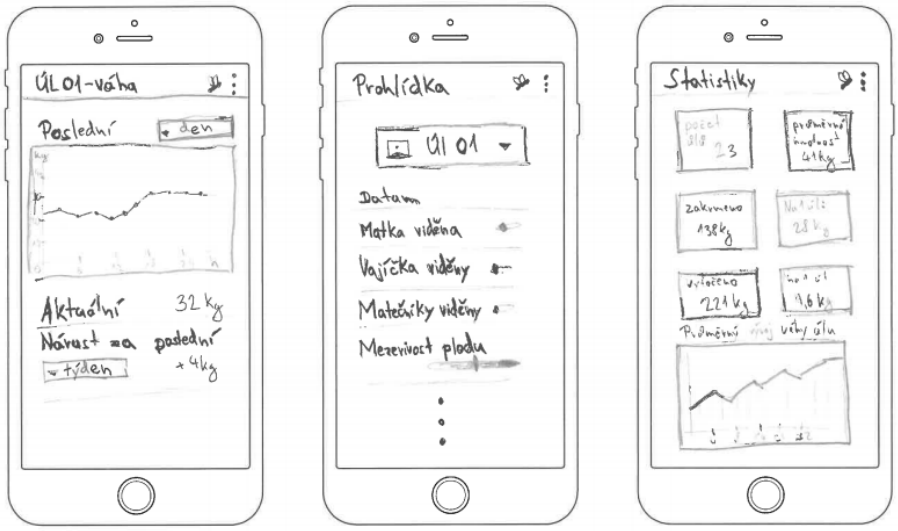
Obrázek 3.3: Skici - možnosti rozložení hlavní obrazovky.

Jako možnost hlavní obrazovky jsem zvolil třetí návrh (vpravo), kde budou tři základní tlačítka pro zobrazení vah, přidání položky a zobrazení statistik. Pod nimi poté seznam stanovišť se základními informacemi.

Další skicy 3.4 obsahují rychlý návrh rozložení obrazovky detailu stanoviště, úlu a matky. Obrazovka zobrazující detail stanoviště bude obsahovat informace o stanovišti a seznam všech úlů nacházejících se na stanovišti se základními informacemi (síla včelstva, váha). Detail úlu poté obsahuje veškeré informace o úlu a poskytuje možnost zobrazit si všechny prohlídky, medobraní, krmení nebo podrobnosti o matce. Poslední obrazovka zobrazuje detailní podrobnosti o matce.

Detail váhy na návrhu 3.5 bude zahrnovat jednoduchý graf s vývojem váhy úlu, aktuální váhu a nárůst za určité období. U přidání prohlídky bude výběr úlu, to ale pouze v případě, že uživatel nezadá přidání prohlídky přímo z detailu konkrétního úlu, dále pak všechny informace k zaznamenání u prohlídky. Obrazovky jako přidání krmení, medobraní nebo léčení budou podobné. Na obrazovce "statistiky" budou vypsány počty úlů, vývoj vah úlů v grafu, množství celkového vytočeného medu nebo přidaného krmiva.

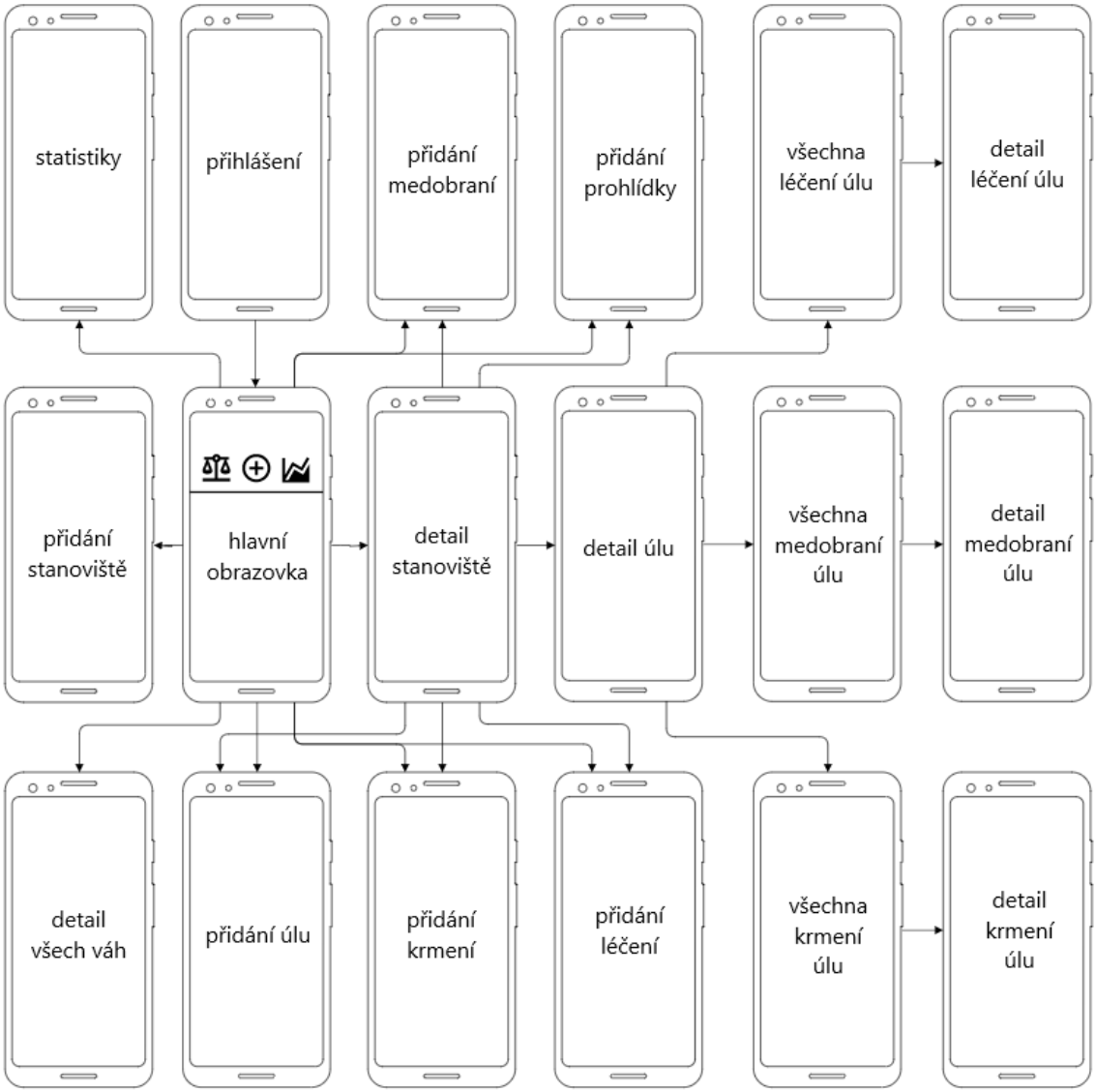
Obrázek 3.4: Skici obrazovek zleva: stanoviště, detail úlu, detail matky.



Obrázek 3.5: Skici obrazovek zleva: detail váhy, přidání prohlídky, zobrazení statistik.

3.4.2 Vztahy mezi jednotlivými obrazovkami

Pro lepší představu vztahů mezi jednotlivými obrazovkami jsem sestrojil diagram obsahující všechny obrazovky výsledné aplikace a vztahy mezi nimi 3.6. Vztah vyobrazuje přechod z jedné obrazovky na druhou. U všech přechodů platí také zpětná návratnost tak, že pokud se uživatel může dostat z obrazovky A do obrazovky B, může se také navrátit z obrazovky B do obrazovky A.



Obrázek 3.6: Zjednodušený mockup aplikace.

3.5 Návrh prototypu úlové váhy

Prototyp chytré úlové váhy je součástí práce právě proto, aby bylo možné reálně simulovat, jak by taková úlová váha mohla s mobilní aplikací komunikovat. Není proto nutné sestavit plně funkční úlovou váhu. Postačí nám zařízení, které bude v pevně daných intervalech za pomocí bezdrátové sítě odesílat informace o váze klidně i náhodně vygenerované, ale v mezích reálných hodnot.

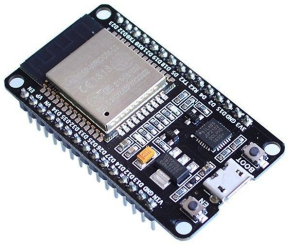
3.5.1 Hardware

Nejprve je nutné zvolit vybrané komponenty, z kterých bude možné prototyp úlové váhy sestrojit. V řešení nezahrnuji konstrukci, která by váhu tvořila, ani tenzometry pro měření

váhy. Účelem je pouze váhu simulovat a k tomu nám postačí mikrokontroler s napájením, který bude schopný posílat nezávisle data.

ESP32

Je vývojová deska 3.7, skvěle se hodící pro tvorbu IoT zařízení. Je to nástupce vývojové desky ESP8266, který má větší výkon, ale i nové funkce, mezi kterými uvítáme právě zabudovanou Wi-Fi, po které bude naše váha komunikovat, ale také zabudovaný teplotní senzor, díky kterému můžeme znát přesnou aktuální teplotu na stanovišti. Procesor obsahuje dvě výpočetní jádra a SRAM paměť má velikost 512 kB. Vstupně-výstupních portů (GPIO) je celkem 36 a oproti předchůdci byly rozšířeny také počty pinů s podporou sběrnic SPI, I2C a UART [15].



Obrázek 3.7: Vývojová deska ESP32[1]

Na vývojové desce poběží program, který bude simulovat měření váhy úlu a odesílat data na server.

Komunikace vývojové desky s aplikací

Nelehkou úlohou při návrhu úlové váhy je zvolení sítě, díky které bude vývojová deska s aplikací komunikovat. Hlavním problémem při výběru sítě je její dostupnost. Stanoviště včelstev se totiž často nachází v přírodě, ať už na loukách nebo v lesích, kde pokrytí signálem ať už jakékoliv sítě nebývá tak dobré, jako ve městech. Při návrhu bylo zohledněno hned několik možných sítí pro komunikaci.

Jako první možnost se nabízí použití sítě **LoRa**. LoRa je mimo jiné označení bezdrátové sítě založené na LPWAN (Low Power Wide Area Network) technologii. Umožňuje jednoduchou a energeticky nízko náročnou komunikaci s velkým dosahem. Pro potřeby úlové váhy se zdá být ideální, takové komunikace by se dalo dosáhnout například za pomocí vývojové desky ESP32 přizpůsobenou pro komunikaci v rámci LoRa sítě. Alternativou použití LoRa sítě může být například další IoT síť pod názvem Sigfox.

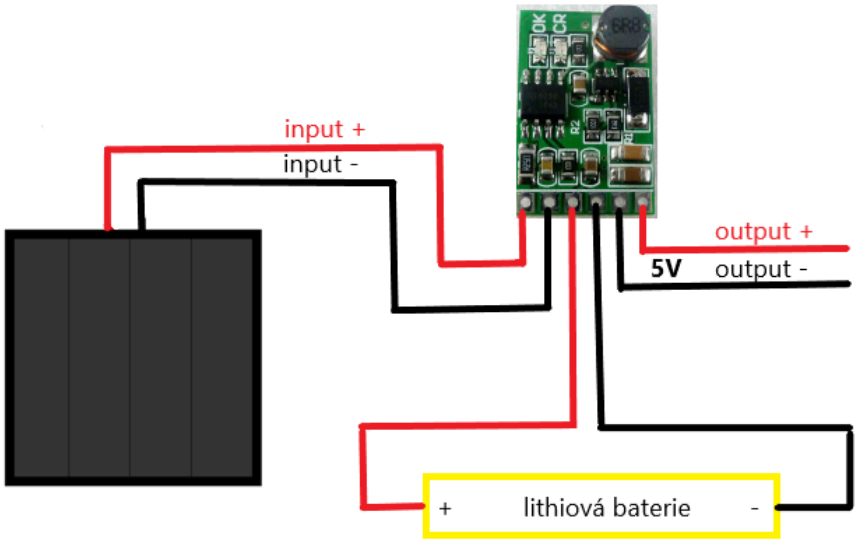
Pro komunikaci úlové váhy může být také využito některé z mobilních datových sítí, například **GPRS**, celým názvem General Packet Radio Service. Jedná se o mobilní datovou síť fungující na principu přepojování paketů a dynamického využívání neobsazených kanálů, které následně sdílí více uživatelů. Stejně jako u LoRy lze v rámci této sítě komunikovat s vývojovou desku přizpůsobenou pro tento typ komunikace, v tomto případě navíc s platnou SIM kartou.

Jako poslední se naskýtá použití Wi-Fi sítě. Ta narozdíl od předchozích sítí nemá pokrytí po celé ČR, ale zato je nejméně finančně náročná. Taková váha by se dala použít pouze na včelstva na zahradách nebo střechách domů, kde se nachází pokrytí Wi-Fi sítě, na kterou se může zařízení připojit. Pro vytvoření prototypu úlové váhy však bohatě postačí.

Napájení vývojové desky

Napájení musí být zajištěno bez přístupu do elektrické sítě, právě kvůli její nedostupnosti v přírodě. Pro maximální výdrž jsem zvolil kombinaci solárního panelu a lithiové baterie jako napájení. Napájecí systém se bude skládat ze 3 komponentů:

- Solární panel — venkovní, vodě odolný.
- Dobíjecí baterie — Lithium Li-ion 18650, 3.7 V a 12000 mAh.
- Modul pro napájení a dobíjení baterie — 5 V UPS Power Diy Board Charger Step-up DC/DC Converter Module.



Obrázek 3.8: Schéma zapojení systému napájení.

Výstup napájecího systému bude napojen na VIN pin vývojové desky ESP32. Ta má již zabudovaný regulátor, který zkonvertuje napětí z 5 V na 3.3 V. Na obrázku 3.8 lze vidět schéma zapojení těchto komponent.

3.5.2 Software

Důležitou součástí úlové váhy je také software. Naskýtá se zde hned několik problému, které je nutné ze softwarového hlediska vyřešit. Prvním z nich je aplikace na samotné vývojové desce, která bude generovat data a odesílat je na cloud. Druhým je, jak tato data dostat z cloudu do databáze mobilní aplikace. A jako poslední je nutné vyřešit, jak propojit data z konkrétní váhy s konkrétním úlem.

Arduino

Je open source hardwarová a softwarová společnost, projekt i komunita, která navrhuje a vyrábí digitální vývojové mikrokontroléry, příslušenství k nim, ale také software pro vývoj na nejen těchto mikrokontrolérech [21]. Software úlové váhy bude vyvíjen právě ve vývojovém prostředí Arduino IDE, které umožňuje jedním kliknutím kompilovat a nahrát kód přímo na vývojovou desku. Navíc poskytne také přístup ke konzolovým výpisům.

Jazyk C

C je imperativní procedurální jazyk. Byl navržen tak, aby poskytoval nízkoúrovňový přístup k paměťovým a jazykovým konstruktům, které efektivně mapují strojové instrukce. To vše s minimální podporou běhového prostředí, skvěle se tak hodí pro nízkoenergeticky náročné programy, jako ten pro chod úlové váhy.

Program nahraný na vývojovou desku ESP32 bude tedy napsán v jazyce C. Jelikož jazyk C sám o sobě neřeší alokaci a dealokaci paměti, v programu bude potřeba dbát na správnou dealokaci všech použitých zdrojů. Je totiž potřeba, aby program běžel bez přerušení a byl zajištěn správný chod úlové váhy.

Kapitola 4

Použité technologie

Pro lepší pochopení platforem, pro které slouží mobilní aplikace Deník včelaře, se v následující kapitole podíváme blíže na jejich operační systémy. Kapitola je dále zaměřena na použité technologie v rámci vývoje týkající se jak mobilní aplikace, tak serverové části i úlové váhy. Seznámíme se se zvolenou aplikační platformou pro vývoj multiplatformních mobilních aplikací Xamarin, frameworkem Xamarin.forms, implementačním jazykem C# a také značkovacím jazykem XAML, který slouží pro sestavení grafického rozhraní aplikace.

4.1 Mobilní platformy

Pojem mobilní platforma se užívá ve spojitosti s hardwarovým a softwarovým prostředím pro mobilní telefony, tablety a další přenosná zařízení. Ve světe mobilních telefonů a tabletů dominují Apple s operačním systémem iOS a Google s operačním systémem Android, tento operační systém využívají i jiné firmy. K březnu 2020 si tyto mobilní platformy rozdělují světový trh mobilních telefonů procentuálně 72,26 % pro Android a 27,03 % pro iOS [5]. Poměr ostatních mobilních platforem na světovém trhu je zanedbatelný. Ještě nedávno měly podíl na trhu také Windows Phone nebo BlackBerry, v současné době již tyto mobilní platformy nejsou téměř používané.

Platformy se vzájemně liší rozhraním operačního systému pro vývojáře mobilních aplikací, funkcemi poskytovanými operačním systémem uživateli i vzhledem a prostředím operačního systému. V neposlední řadě ale také aplikacemi třetích stran, které jsou na určité platformě k dispozici.

4.1.1 Android

Android je v současné době nejpoužívanější operační systém v oblasti mobilních telefonů. Byl vytvořený firmou Android, Inc., v roce 2005 jej koupila společnost Google, která do dnešního dne pokračuje v jeho vývoji. Tento operační systém je tak rozsáhlý také z toho důvodu, že je to open-source software a mohou jej využívat i ostatní výrobci mobilních telefonů, nejenom společnost Google. K jeho rozsáhlosti přispívá také fakt, že se nejedná o platformu jenom pro mobilní telefony, ale také pro ostatní přenosná zařízení jako jsou tablety, chytré hodinky nebo například televize.

Operační systém Android běží na bázi Linuxového jádra [19]. Největší jeho výhodou a zároveň i nevýhodou je jeho otevřenost. Pro koncové zákazníky je tento operační systém nabídnut na opravdu široké paletě zařízení různých značek a specifikací, zákazníci si tedy mají z čeho vybírat. Výrobci mobilních telefonů si také mohou různě upravovat chování a

vzhled systému, neboli vytvářet nástavby. To sebou nese značné nevýhody, jelikož systém není přímo optimalizovaný na konkrétní hardware. Navíc s každou novou aktualizací musí jednotliví výrobci mobilních telefonů provést úpravu na svoji nástavbu a až poté mohou aktualizaci doručit koncovým uživatelům.

4.1.2 iOS

Oproti tomu iOS je operační systém určený výhradně pro zařízení společnosti Apple, Inc.. Ještě pod původním názvem iPhone OS byl vyvinut na základě operačního systému pro počítače Macintosh macOS [20].

Jeho hlavní dominantou je uzavřenost. Není jej možné přizpůsobit pomocí nástaveb, ani do něj instalovat jiné aplikace než schválené Applem a publikované v obchodě App Store. Jeho uzavřenost výskytu pouze na zařízeních společnosti Apple přináší také výhody v rychlosti doručení aktualizací koncovým zákazníkům, tato výhoda je nesporná hlavně z důvodu rychlosti nasazení bezpečnostních záplat.

4.2 Multiplatformí vývoj mobilních aplikací

Pokud chceme, aby se naše aplikace dostala k co nejvíce uživatelům, vývoj pro více platforem je v dnešním světě nezbytný. Nicméně vývoj mobilní aplikace pro rozdílné platformy není jednoduchý, vývojář musí vytvořit aplikaci několikrát, s různými technologiemi a různými nástroji. To spotřebuje mnoho času a úsilí, hlavně pokud takové aplikace vyvíjí jeden člověk. Zde však nastupuje multiplatformní vývoj mobilních aplikací, který tyto problémy řeší. Multiplatformní vývoj je proces tvorby mobilní aplikace, která může být vydána pro více platforem, s použitím stejného kódu. Umožňuje tedy vytvořit jednu aplikaci spustitelnou na více platformách namísto vývoje několika aplikací pro každou platformu zvlášť. V mém případě se jedná o platformy Android a iOS.

Mobilní aplikace můžeme rozdělit do několika kategorií z hlediska vývoje a funkčnosti. Tyto kategorie si v následující části trochu přiblížíme.

Nativní aplikace

Nativní mobilní aplikace jsou vyvíjeny tak, aby fungovaly pouze na specifické platformě nebo operačním systému. U programování aplikací pro platformu iOS vývojaří využívají programovacího jazyku swift nebo objevtive-c a vývojové prostředí xcode, zatímco pro Android, vývojáři používají Kotlin nebo Javu ve vývojovém prostředí Android Studio.

Jejich značná výhoda je, že vývojáři nativních aplikací mají jako první přístup k novým funkcím specifických platforem. Mohou tak rychle reagovat na změny v systému a udržovat své aplikace aktuální.

Hybridní multiplatformní aplikace

Hybridní mobilní aplikace jsou v podstatě webové aplikace, které jsou zaobaleny do nativního kontejneru využívajícího vestavěné prohlížeče jednotlivých platforem. Vývojář tedy vytvoří jednu webovou aplikaci a tu pak zaobalí do nativního kontejneru prohlížeče pro každou platformu. Výsledná aplikace tedy působí dojmem nativní aplikace a může být instalována na zařízení. Navíc pro její spuštění není nutné připojení k internetu, jelikož jsou spuštěny v kontejneru vestavěného prohlížeče a kód webové aplikace je součástí této aplikace.

Hlavním benefitem je sdílený kód, rychlý vývoj a konzistentní UI napříč platformami. Pro jejich vývoj se převážně využívá technologií HTML5, Javascript a CSS [6].

Nativní multiplatformní aplikace

Nativní multiplatformní aplikace fungují na principu sdíleného kódu, který je poté přeložen do nativního kódu cílové platformy. Stejně jako hybridní vývoj nám umožňuje publikovat aplikace s minimálním úsilím na více platforem. Jedná se o perfektní kombinaci hybridních a nativních aplikací, kdy na jednu stranu není potřeba vyvíjet více aplikaci pro každou platformu a na straně druhé se jedná o aplikace se zlepšeným výkonem podobným výkonu nativních aplikací. Pro vývoj nativních multiplatformních aplikací se využívá technologií React Native, Xamarin nebo nově také Flutter. Vývoj nativní multiplatformní aplikace jsem pro její nesporné výhody zvolil také v rámci vývoje mobilní aplikace Včelařského deníku. Mezi jakými technologiemi pro tento vývoj jsem vybíral, se podíváme v následující kapitole.

4.3 Srovnání technologií pro vývoj multiplatformních mobilních aplikací

Při vývoji multiplatformních mobilních aplikací není snadné zvolit mezi jednotlivými technologiemi pro vývoj. V mnoha ohledech poskytují stejné možnosti, avšak v mnoha se také liší. Každá technologie má své výhody a nevýhody, v následující sekci si přiblížíme nejpoužívanější technologie pro tento druh vývoje, Xamarin záměrně není zahrnut, jelikož je detailněji přiblížen v další sekci.

React Native

React Native je open-source mobilní aplikační framework vytvořen společností Facebook. Umožňuje současný vývoj jedné aplikace pro Android a iOS. Kombinuje nativní vývoj s Reactem, Javascriptovou knihovnou pro tvorbu uživatelského prostředí [8]. React Native tak umožňuje vývoj mobilních aplikací v jazyce Javascript, ty jsou poté přeloženy do nativních aplikaci, které jsou téměř identické s aplikacemi napsány za pomocí nástrojů pro nativní vývoj.

Jeho výhody můžeme najít v rychlosti výsledné aplikace, použití nativních UI komponent a možností úpravy vzhledu aplikace za běhu při vývoji, tzv. Hot reloading. Nehodí se však na vývoj aplikací s animacemi a složitými přechody, kde ztrácí na výkonu. To se projevuje i na navigaci v React Native aplikacích, která není tak plynulá.

Flutter

Flutter je aplikační vývojový nástroj pro vývoj multiplatformních aplikací, vytvořen společností Google. Převážně využíván na tvorbu mobilních aplikací pro Android a iOS. Nově se však chystá také rozšíření pro web a desktop. Vývojář tak bude schopný napsat aplikaci rovnou pro čtyři platformy s jedním sdíleným kódem. Aplikace vyvíjeny za pomocí technologie Flutter jsou psány v jazyce Dart. Základním stavebním kamenem pro tvorbu uživatelského rozhraní je widget. Za pomoci widgetu lze deklarovat téměř každý grafický prvek uživatelského rozhraní [4].

Flutter také podporuje hot realoading, který je navíc opravdu plynulý. Díky widgetům je snadné přizpůsobit si jakýkoliv prvek uživatelského rozhraní daným potřebám. Nevýhodou

je větší velikost výsledných aplikací a také se jedná stále o nový nástroj a netěší se silné vývojářské základně. Poslední dobou však rychle nabírá na popularitě.

4.4 Xamarin

Xamarin je open source aplikační platforma od Microsoftu pro tvorbu moderních a výkonných aplikací pro iOS, Android i Windows za pomocí C# .NET a XAML [17]. Umožňuje také vývoj pro tvOS, macOS a watchOS, a to vše s podporou nativních funkcí [12].

.NET je vývojářská platforma tvořená nástroji, programovacími jazyky a knihovnami pro vývoj aplikací. Základní platforma poskytuje komponenty k různým typům aplikací a přídavné frameworky, jako Xamarin, rozšiřují .NET o komponenty pro vývoj aplikaci pro další cílové platformy. Zde je pár komponent, které sebou přináší samotná .NET platforma:

- **C#** programovací jazyk a jeho kompilátory.
- **Základní knihovny** pro práci s datovými typy jako jsou textové řetězce, data, přístup k čtení a zápisu u souborů a další.
- **Editory a nástroje** pro Windows, Linux, macOS a Docker.

Jak již bylo řečeno, Xamarin dále rozšiřuje .NET platformu o nástroje a knihovny specifické pro vývoj aplikací na iOS, Android, Windows, macOS a další. Zde je pár komponent, o které Xamarin rozšiřuje .NET platformu:

- **Základní framework pro přístup k nativním funkcím.**
- **Rozšiřující značkovací jazyk** známý také jako XAML. Pro tvorbu uživatelského rozhraní.
- **Knihovny specifické pro platformu**, které poskytují přístup k aplikačním rozhraním společností jako Google, Apple nebo Facebook a další bohaté možnosti.
- **Rozšíření pro editor** k zvýraznění syntaxe, napovídání k automatickému dokončování kódu, designové nástroje a další funkcionality specifické pro vývoj mobilních aplikací.

Díky tomu, že Xamarin rozšiřuje platformu .NET, je možné při vývoji použít veškeré balíčky a knihovny dostupné všem .NET vývojářům. Za pomocí nástroje pro zprávu balíčku pro .NET, NuGet[1], je jich dostupných více než 200 000, také je možné si vytvořit své vlastní a sdílet je s ostatními vývojáři.

Kromě výše zmiňovaných výhod technologie Xamarin jsem měl i několik dalších důvodů, proč zrovna upřednostnit Xamarin před vývojem za pomocí ostatních technologií pro vývoj multiplatformních mobilních aplikací:

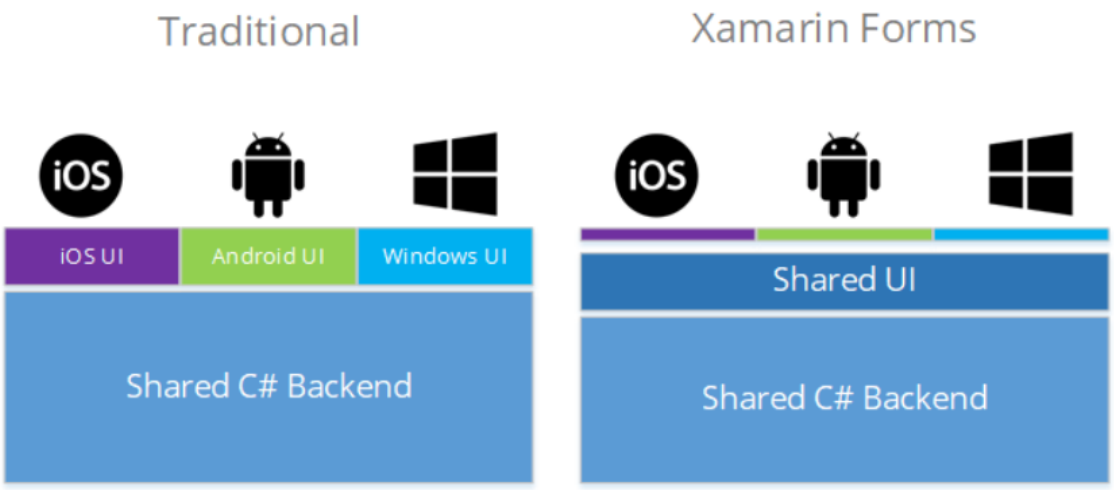
- **Výkon**: Xamarin aplikace jsou známe pro svou úroveň výkonu podobou nativním aplikacím.
- **Vývojové prostředí**: Microsoft Visual Studio, .Net a C# jsou všechny potřebné nástroje pro vývoj mobilních aplikací.

[1] https://www.nuget.org/

- **Windows aplikace**: Xamarin mi poskytuje možnost v budoucnu vydat výslednou aplikaci i pro desktopové počítače s operačním systémem Windows, jednoduše přidáním UWP projektu.
- **Zkušenosti**: Ačkoli framework Xamarin používám poprvé, s vývojovým prostředím Visual studio, technologií .Net a jazykem C# mám již zkušenosti. Ať už z předmětu IW5 nebo ICS, tak ze soukromých projektů.

4.4.1 Xamarin.forms

Je open-source framework pro vývoj aplikací se sdílenou logikou i uživatelským rozhraním pro iOS, Android i Windows. Rozšiřuje Xamarin o nástroje a knihovny, které takový vývoj umožňují. Xamarin.Forms však nepřináší jen sdílené uživatelské rozhraní napříč platformami, ale také všechno, co potřebujeme k takovému vývoji mobilních aplikací. Zahrnuje tedy také sdílenou navigaci pro všechny platformy, aplikační rozhraní pro animace, dependency service[2], centrum zpráv a další. Na obrázku 4.1 je znázorněn rozdíl mezi vývojem v klasické Xamarin platformě a vývojem v Xamarin.forms.



Obrázek 4.1: Graficky znázorněný rozdíl mezi Xamarin a Xamarin.forms[3].

Xamarin.forms umožňuje deklarativní vývoj uživatelského rozhraní za pomoci značkovacího jazyka XAML, nicméně umožňuje také vývoj stejného uživatelského rozhraní za pomocí jazyka C#, je jen na konkrétním vývojáři, kterou cestu si vybere. Standardní je použít XAML [23].

4.4.2 C#

Je moderní, objektově orientovaný a typovaný programovací jazyk. Jeho kořeny sahají k rodině C jazyků, jeho syntaxe bude povědomá C, C++, Java a Javascript vývojářům.

[2] https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/app-fundamentals/dependency-service/introduction

```
using System;

class Hello
{
    static void Main(){
        Console.WriteLine("Hello, World!");
    }
}
```

Výpis 4.1: Ukázka v jazyce C#.

Funkce jazyka C# pomáhají při konstrukci robustních a odolných aplikací. Garbage collector automaticky uvolňuje paměť obsazenou nedostupnými, již nepoužitými objekty. Zpracování výjimek poskytuje strukturovaný a rozšiřitelný přístup k detekci a obnovení z chyb. Typově bezpečný design jazyka zase znemožňuje čtení z neinicializovaných proměnných, indexování polí mimo jejich šířku nebo provádět netypované přiřazení do proměnné [9]. V ukázce kódu 4.1 je vidět jednoduchý konzolový program napsaný v jazyce C#, který vypíše do konzole hlášku.

4.4.3 XAML

XAML je zkratka pro eXtensible Application Markup Language, česky řečeno rozšiřitelný značkovací jazyk pro aplikace. Tento jazyk je variantou XML od společnosti Microsoft, v jeho ukázce 4.2 je vidět jednoduchá deklarace tlačítka. Byl vytvořen pro tvorbu uživatelského rozhraní aplikací [14]. S jazykem XAML je psaní kódu uživatelského rozhraní podobné s psaním webu v HTML.

```
<StackPanel>
    <Button Content="Click Me"/>
</Stackpanel>
```

Výpis 4.2: Deklarace tlačítka uvnitř stack panelu v jazyce XAML.

S tímto jazykem je možné vytvořit prvky uživatelského rozhraní a poté oddělit definici uživatelského rozhraní od logiky aplikace za pomocí částečných definic tříd (partial class). Každý takový prvek je tedy tvořen dvěma soubory, jedním s koncovkou `.xaml`, kde

1. Soubor s koncovkou `.xaml`, kde je definováno uživatelské rozhraní.
2. S koncouvkou `.cs`, kde je v jazyce C# napsána logika.

Při použití návrhového vzoru MVVM to však úplně neplatí a logika se nachází ve ViewModel třídách.

4.5 ASP.NET

Je open-source webový framework vytvořený Microsoftem. Slouží pro tvorbu moderních webových aplikací a služeb s technologií .NET. ASP.NET je multiplatformní a běží na Windows, macOS, Linuxu i Dockeru. Stejně jako Xamarin, rozšiřuje základní .NET platformu o další nástroje a knihovny, nyní však specifické pro vývoj webových aplikací:

- **Základní framework pro zpracování webových požadavků v C# a F#.**
- **Syntaxi pro šablony webových stránek** známou jako Razer, k budování dynamických webových stránek s jazykem C#.
- **Knihovny pro webové návrhové vzory** jako Model View Controller (MVC).
- **Authentikační systém.**
- **Rozšíření editoru** pro zvýraznění syntaxe, napovídání k dokončení kódu a další funkce specifické pro vývoj webových aplikací.

S ASP.NET je možné budovat mnoho typů webových aplikací včetně webových stránek, REST APIs, mikroslužeb a dalších. V rámci této bakalářské práce bylo využito ASP.NET aplikace hlavně k vytvoření REST API[4] pro přístup k datům z databáze.

4.6 Azure

Microsoft Azure je rodina cloudových služeb vytvořena společností Microsoft. Slouží pro vývoj, testování, nasazení a správu aplikací a služeb skrze datová centra spravována Microsoftem. Jejich služby se dělí do několika skupin:

- Software as a service (SaaS) — je způsob nasazení software, který je hostován na Azure serverech a poskytován zákazníkům přes internet.
- Platform as a service (PaaS) — zákazníkům je poskytnut přístup k využívání platforem jako jsou operační systémy, vývojové nástroje, systémy pro správu databází, testování a další. Tyto platformy jsou navíc v maximální možné míře provázány mezi sebou přímo poskytovatelem.
- Infrastructure as a service (IaaS) — nabízí funkcionalitu systémové integrace, umožňuje tak přenášet data on-demand napříč systémy.

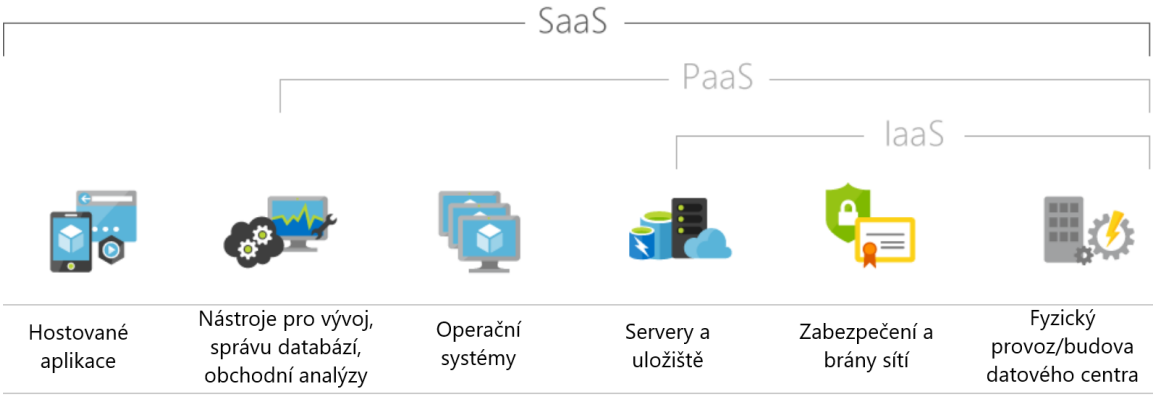
Přesnější představu o dělení cloudových služeb lze získat z obrázku 4.2. Azure služeb bude v maximální možné míře využito v rámci studentského předplatného Azure for Students, které máme dostupné v rámci emailové adresy naší školy.

App service

Azure App Service je platforma sloužící pro hostování webových aplikací a služeb. Patří do skupiny PaaS služeb. Umožňuje hostování webových aplikací běžících na různých frameworcích a napsaných v různých programovacích jazycích (.NET, .NET Core, node.js, PHP, Python, Ruby a Java) [1]. Původně tato služba byla poskytována pod názvem Windows Azure Web Sites, poté byla přejmenována na Microsoft Azure Web Sites a nakonec v roce 2015 dostala název App Service, starší zákazníci ji tedy můžou znát pod jiným názvem.

Pro účely této bakalářské práce poslouží k hostování webové aplikace s REST API napsané v ASP.NET. App Service navíc poskytne zabudovanou podporu authentizace a autorizace uživatelů k API za pomocí přístupových webových JSON tokenů.

[4] https://dotnet.microsoft.com/apps/aspnet/apis



Obrázek 4.2: Dělení cloudových služeb Microsoft Azure[5].

SQL Databáze

Pro ukládání uživatelských dat je potřeba zabezpečit také bezpečné cloudové uložiště. K tomu skvěle poslouží služba Azure SQL Database, která poskytuje výkonné, dostupné uložiště dat. Azure SQL Database také patří do skupiny PaaS služeb. Data uživatelů tak budou bezpečně uložena v databázi na SQL serverech Microsoftu.

Active Directory B2C

Azure Active Directory B2C je služba zabezpečující správu identit zákazníků a přístup k aplikacím. Tuto službu lze také využít v mobilních aplikacích, uživatel se tak může jednoduše registrovat, při registraci ověřit emailovou adresu, přihlašovat, ale také získat zapomenuté heslo nebo měnit údaje svého účtu. Vše lze také přizpůsobit vzhledu a potřebám specifické aplikace [2].

Kapitola 5

Implementace

Pro úspěšnou implementaci všech systémů bylo nezbytné provést pečlivý návrh a zvolit odpovídající technologie. Tyto kroky byly úspěšně splněny a nyní se můžeme přesunout k samotné implementaci.

Hlavní část této kapitoly zabírá sekce popisující implementaci mobilní aplikace. V této sekci je celý koncept nejen mobilní aplikace, ale také způsob komunikace se serverem a úlovou váhou. Dále možnosti přihlašování a registrace do aplikace, použité architektonické vzory, zajímavé části a překážky samotné implementace v kódu. Další sekce této kapitoly se zabývají implementací a samotným sestavením úlové váhy a nakonec serverové části, která je pro chod celého systému nezbytná.

5.1 Mobilní aplikace

Aby výsledná aplikace mohla pomoci co nejvíce včelařům po celém světě, byl zvolen právě multiplatformní vývoj, a to v technologii Xamarin.Forms. Díky tomu bylo možné dosáhnout jedné implementace aplikace pro obě nejpoužívanější platformy Android a iOS. Všechny texty aplikace jsou navíc přeloženy jak do českého, tak anglického jazyka, v budoucnu není problém přidat jazyky další. Samotná implementace se dělí na dvě hlavní části, a to sestavení grafického rozhraní, neboli to, jak bude aplikace vypadat, a napsání logiky aplikace, to, jak se bude v jednotlivých situacích chovat.

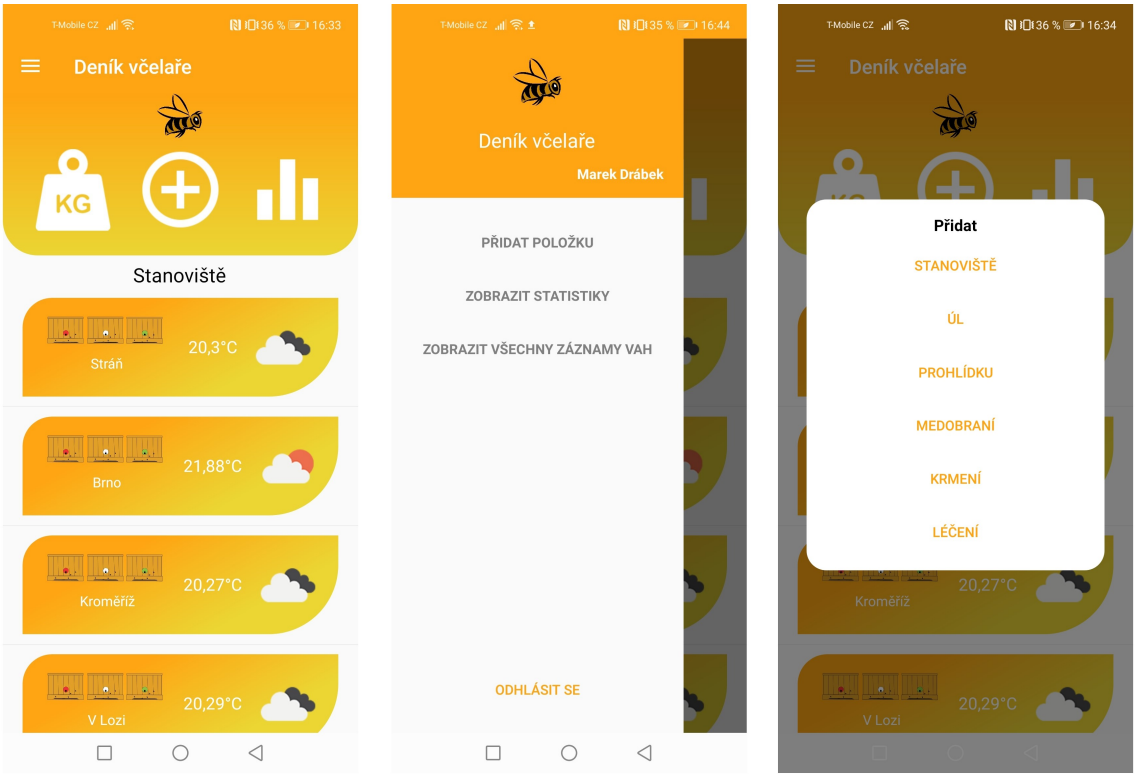
5.1.1 Grafické rozhraní

Prvním úkolem implementace bylo přetavit návrh uživatelského rozhraní do spustitelné aplikace. Jelikož při návrhu nebyly nakresleny všechny obrazovky výsledné aplikace, ale jen náčrt za pomocí skic 3.4.1 těch nejdůležitějších, bylo nutné vyřešit konkrétní vzhled jednotlivých obrazovek v tuto chvíli. Dle návrhu byl pro sestavení uživatelského rozhraní použit značkovací jazyk XAML.

Důležitém aspektem bylo zvolení správných barev aplikace. Barvy by měli odpovídat účelu aplikace, v tomto případě bylo zvolení vhodných barev tedy jednoduché – barvy medu a vosku, tedy oranžová a žlutá. Pro moderní vzhled bylo pro některé prvky využito přechodu těchto dvou barev. Barva písma je poté bílá na oranžovém nebo žlutém pozadí a černá či jemně šedá na bílém pozadí. Výjimku tvoří barva písma některých tlačítek.

Hlavní obrazovka aplikace

Složitou volbou při tvorbě uživatelského rozhraní bylo zvolení rozložení hlavní obrazovky. V návrhu bylo vytvořeno hned několik možných podob tohoto rozložení. Nakonec bylo vybráno rozložení s levým bočním menu, logem s třemi hlavními tlačítky v horní části obrazovky a seznamem všech stanovišť daného uživatele 5.1. V seznamu stanovišť je poté u každého stanoviště ikona, název stanoviště a aktuální počasí na tomto stanovišti.



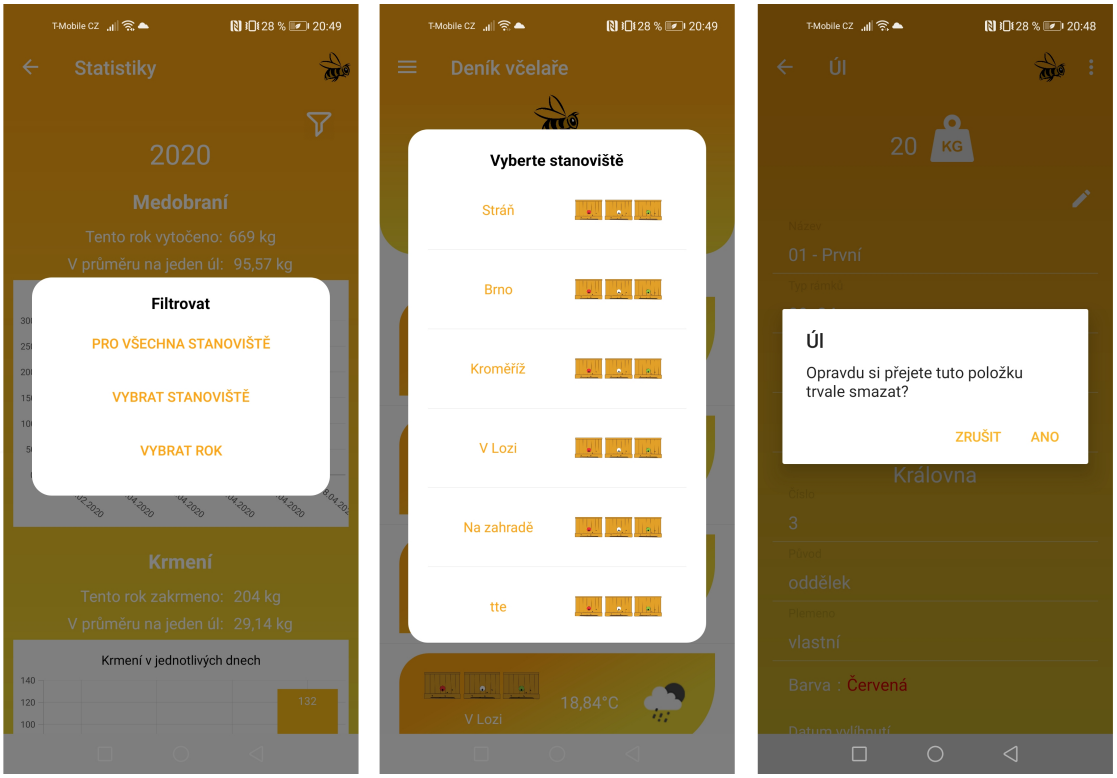
Obrázek 5.1: Ukázka z mobilní aplikace: hlavní obrazovka.

Hlavní obrazovka slouží uživateli jako odrazový můstek pro veškerou práci, kterou bude s aplikací provádět. Při jejím sestavování byl dbáno na to, aby neobsahovala zbytečné rušivé elementy a poskytovala rychlí přístup ke všem případům užití z návrhu.

Dialogová okna

Pro příjemnou uživatelskou zkušenost jsou součástí aplikace dialogová okna. Dialogová okna mají různá využití, slouží například pro rychlý výběr z několika možností. Tato varianta využití dialogového okna se vyskytuje například při výběru stanoviště, úlu na stanovišti, ale také při akci, jakou je přidání záznamu do databáze. Takové využití můžeme vidět například v pravém záznamu obrazovky na obrázku 5.1. Dialogová okna pro výběr mají vždy bílé pozadí a zaoblené rohy. Jsou implementována za pomocí knihovny Rg.Plugins.Popup [1], tato knihovna umožňuje vytvoření stránky, která se graficky chová jako dialogové okno. Další využití dialogových oken pro výběr se nachází v sekci statistik, kde slouží pro výběr filtru, specifického roku nebo například pro výběr stanoviště.

[1] https://github.com/rotorgames/Rg.Plugins.Popup



Obrázek 5.2: Ukázka dialogových oken zleva: výběr filtru, výběr stanoviště a potvrzení smazání úlu.

Dialogová okna jsou také prostředkem pro potvrzování specifických operací, kdy má uživatel možnost operaci potvrdit nebo naopak zrušit. Taková okna jsou použita u operací, které nejdou zpětně vrátit, typicky se jedná o smazání určité položky, kdy je dobré mít jistotu, že uživatel chtěl akci skutečně provést. Vícero použití dialogových oken lze vidět na obrázku 5.2.

Formuláře

Deník ze své podstaty slouží k zaznamenávání informací, k tomu v aplikaci slouží četná formulářová pole, do kterých uživatel může informace zaznamenávat. Formuláře mohou mít různou podobu a mohou sloužit k zaznamenání různých druhů dat. V aplikaci Deník včelaře slouží pro zaznamenávání krátkých textů, dlouhých textů, čísel, datumů, ale také pravdivostních hodnot.

Pro konzistentní vzhled celé aplikace je použito vizuálnosti Material Visual 5.1.1 a jednotného stylu nadefinovaného v Resource Dictionary 5.1.1 souboru `default.xaml`. Stejnou vizuální podobností nejen formulářů, ale i dalších vizuálních prvků aplikace chceme dosáhnout pozitivního přijetí uživatelského rozhraní uživatelem. Pole pro zaznamenání textů a čísel jsou animované tak, že při zaměření na dané pole dojde k plynulému přechodu textu vyjadřujícího typ zaznamenávané hodnoty do horní části, tento efekt by měl také přispět k pozitivnější uživatelské zkušenosti s aplikací.

Docílení příjemné uživatelské zkušenosti u formulářů však nekončí jen u zaručení konzistentního vzhledu. Důležitým faktorem u formulářů je také automatické vyplňování polí. Typickým příkladem automatického vyplňování může být datum u zaznamenání události.

Všechny události jako jsou prohlídky, medobraní, krmení a léčení jsou zaznamenávány vždy s konkrétním datem, toto datum je vždy předvyplněno aktuálním dnem. Dalším takovým příkladem je předem zvolená barva královny 2.1 při přidávání úlu, dle aktuálního roku.

Konvertory

Převodníky hodnoty (angl. value converters, dále jen konvertory) jsou neoddělitelnou součástí grafického rozhraní, dokáží totiž měnit hodnoty vlastností grafických prvků. Konvertory tvoří pomyslný most mezi zdrojem dat a cílem, pokud zdroj a cíl mají jiný datový formát nebo je prostě potřeba jen hodnoty trochu poupravit.

```
<StackLayout
    IsVisible="{Binding Path=Apiary.Weather,
        Converter={StaticResource NullToFalseConverter}}"
    Orientation="Horizontal"
    HorizontalOptions="Center"
    VerticalOptions="Center"
    Grid.Row="0">
    <Label
        Style="{StaticResource TextNormal}"
        HorizontalOptions="Center"
        VerticalOptions="Center"
        FontSize="17"
        Text="{Binding Apiary.Weather.Main.Temperature,
            StringFormat='{}{0}C'}" />
    <Image
        Source="{Binding Path=Apiary.Weather.Weather[0].Icon,
            Converter={StaticResource WeatherIconNameConverter}}"/>
</StackLayout>
```

Výpis 5.1: Panel pro zobrazení počasí s použitím dvou konvertorů.

V aplikaci je použito konvertorů na mnoha místech, typickým příkladem může být zobrazení počasí na stanovišti 5.1. V tomto případě jsou použity konvertory rovnou dva. `WeatherIconNameConverter` pro převod názvu počasí na název ikony reprezentující toto počasí, aby mohla být zobrazena. A `NullToFalseConverter` pro převod počasí na pravdivostní hodnoty, kdy pokud jsou informace o počasí dostupné, vrací `true` a počasí je zobrazeno, ale pokud informace o počasí dostupné nejsou, vrací `false`, a počasí tak zobrazeno není. Vizuální zobrazení fungování těchto konvertorů lze vidět v levé části obrázku 5.3.

Další možným příkladem užití je změna barvy textu na obrazovce "přidat úl". Uživatel v tomto případě vybírá barvu matky a volba se promítá do textu, reprezentujícího název této barvy. Tento text je za pomoci `StringToColorConverter` převeden na barvu tohoto textu. Ukázku lze vidět na pravé obrazovce obrázku 5.3.

Material Visual

Material Design je dogmatický designový systém vytvořený společností Google, který předepisuje velikost, barvu, vzdálenosti a další aspekty vizuálnosti i rozložení aplikací. Material Design byl uveden právě proto, aby bylo popsáno jak by interaktivní uživatelské rozhraní mělo být implementováno k dosažení skvělé použitelnosti a k potěšení uživatelů.

Obrázek 5.3: Ukázka fungování konvertorů při zobrazení informací o počasí na stanovišti, na levé obrazovce jsou informace o počasí dostupné a na pravé ne. V pravé části je ukázka použití konvertoru pro převod textu na barvu.

Xamarin.Forms Material Visual je knihovna, která může být použita k dosažení Material Designu v Xamarin.Forms aplikacích. Material Visual umožňuje tvořit aplikace, které navíc vypadají téměř identicky na obou platformách iOS a Android. Pokud je Material Visual zakomponován do aplikace, podporované grafické prvky adoptují stejný design napříč platformami, což dopomáhá k vytvoření jednotného vzhledu a dojmu [13].

Použití stejného designu na obou cílových platformách přináší své velké výhody, ale také nevýhody. Nevýhodou může být zvyklost uživatelů dané platformy na design specifický pro tuto platformu. Velkou výhodou ale je, že není potřeba vytvářet dva odlišné návrhy pro každou platformu a také není nutné rozhodovat, zda určité grafické prvky je potřeba udělat spíše Android nebo spíše iOS cestou, aby se vyhovělo konvencím. Vytvoření jednotného designu napříč všemi mobilními zařízeními je také klíčovým bodem v udržování konzistentního uživatelského zážitku pro všechny uživatele napříč platformami.

Resource Dictionary

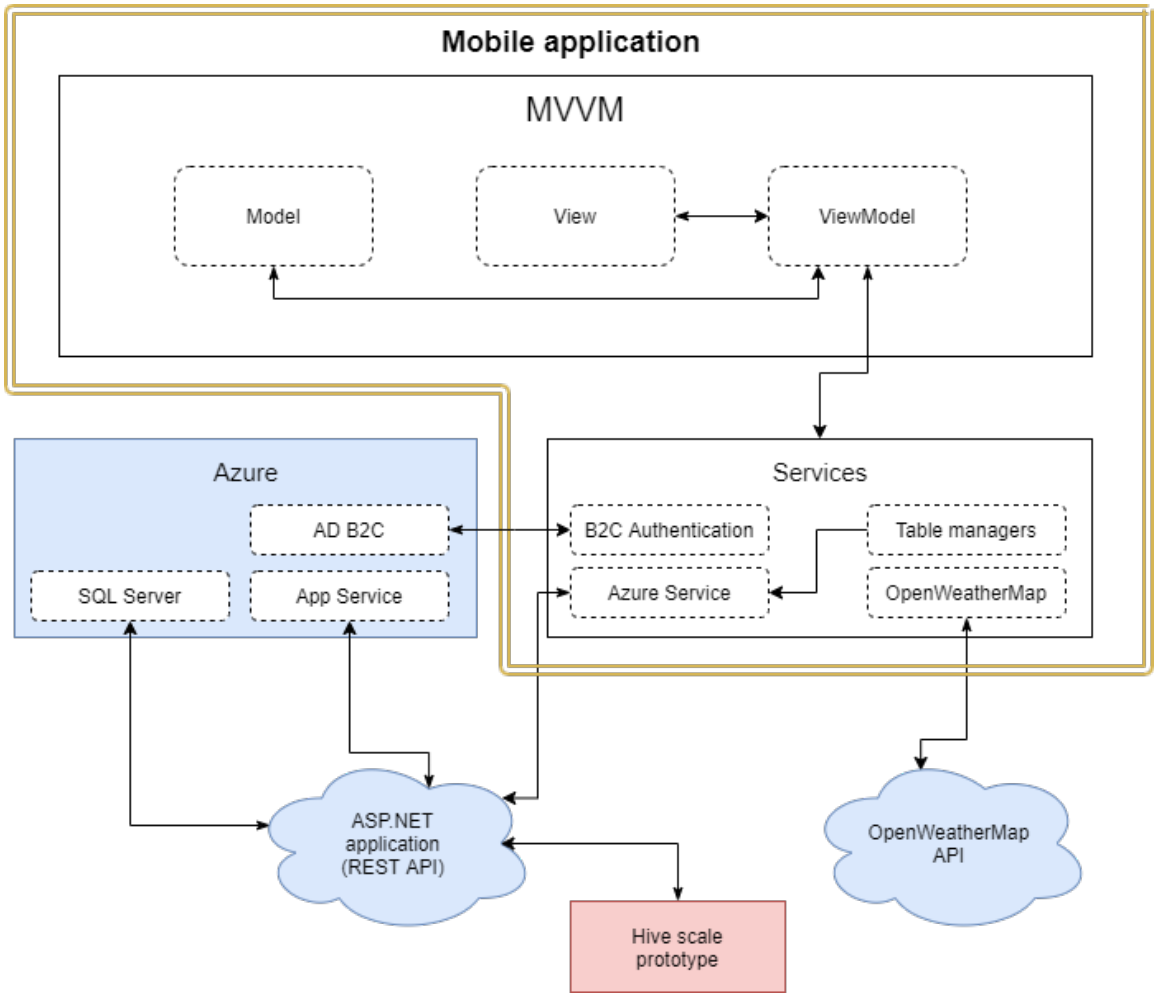
K úplnému zajištění konzistentního vzhledu grafických prvků v aplikaci poslouží sdílené resource dictionary. Jsou to slovníky, které usnadňují sdílení konstant a stylů pro použití v jazyce XAML. V resource dictionary tak lze definovat styl grafického prvku a ten pak sdílet napříč aplikací.

V práci jsou použity hlavně na definování sdíleného stylu tlačítek, vstupních polí, editorů a písma. Tyto předdefinované styly zároveň usnadňují práci v tom, že ke každému prvku poté není nutné vypisovat všechny grafické vlastnosti, ale většinu vlastností jako je

barva pozadí, barva písma a další adoptuje z předdefinovaného stylu. Definování sdílených stylů však není jedinou využitelností resource dictionary v aplikaci. Jsou použity také na definování barev, aby bylo docíleno stejného barevného rozložení na všech obrazovkách.

5.1.2 Architektura

Je na čase se přesunout od grafického rozhraní na samotnou logiku, či architekturu nejen mobilní aplikace, ale celého systému, který je součástí práce. Představení celého systému je nezbytné pro pochopení komunikace aplikace s databází, úlovou váhou, ale také s autentizačním serverem. Popsat celou architekturu systému pouze slovy by bylo pro pochopení čtenáře složité, v následující sekci tak bude popsána architektura podle grafu 5.4, dále jen graf, který zjednodušeně tento systém reprezentuje.



Obrázek 5.4: Graf zobrazující komunikaci jednotlivých prvků systému spojeného s mobilní aplikací Deník včelaře.

Mobilní aplikace je postavena na architektonickém vzoru MVVM 5.1.2, kromě toho jsou na grafu jako součást mobilní aplikace vyobrazeny služby, které slouží ke komunikaci aplikace s okolním světem. Tyto služby patří do okruhu `ViewModel`, ale pro lepší pochopení komunikace mezi jednotlivými komponenty systému byly vyobrazeny v samostatné části.

Služba `B2C Authentication` je blíže popsána v sekci rozebírající přihlašování 5.1.4, ostatní služby budou blíže přiblíženy nyní.

Azure Service

Je služba reprezentována třídou `AzureService.cs` implementovanou jako singleton , která se stará o veškerou komunikaci s ASP.NET aplikací hostovanou na Azuru v App Service 4.6, ale také o inicializaci offline uložiště pro ukládání dat uživatele přímo na mobilním telefonu. Veškerá komunikace s aplikací na Azure probíhá za pomocí instance `MobileServiceClient` třídy, která je součástí knihovny Microsoft.Azure.Mobile.Client[2]. Za pomocí této třídy je možné se dotazovat na data z databáze, synchronizovat data uložená na zařízení s daty uloženými na serveru, ale také se autentizovat na serveru za pomocí přístupových tokenů.

Table managers

Pro každou tabulku z databáze existuje v aplikaci jedna třída, která umožňuje práci s daty dané tabulky. Tyto třídy se obecně nazývají table managers a slouží k inicializaci dané tabulky, offline synchronizaci dat tak, že nejdříve odešlou lokální změny na server a poté stáhnou změny ze serveru do lokálního uložiště. Zajišťuje také ale řešení konfliktů, v případě, že se nepodaří nahrát určitou změnu v databázi na server nebo naopak. Tato synchronizace probíhá při jakékoliv změně dat v databázi, ale také při získávání dat z databáze. Komunikace se serverem probíhá za pomocí třídy `AzureService.cs`, konkrétně s instancí `MobileServiceClient`.

Hlavním účelem těchto tříd však je umožnění práce s databází pro jednotlivé tabulky. Každá třída obsahuje metody `AddItem`, `GetItems`, `UpdateItem`, `DeleteItem`, kde `Item` představuje v každé třídě název dané tabulky.

B2C Authentication

Je služba tvořená třídou `B2CAuthenticationService.cs`, která zajišťuje veškerou logiku a třídou `B2CConstants.cs`, ve které se nachází nezbytné konstanty k připojení na Azure Active Directory B2C 4.6. Podstatou této služby je zajišťovat uživateli přístup k přihlašování, vytváření účtu, ale také umožnění přístupu k znovuobnovení zapomenutého hesla.

Komunikace probíhá tak, že po kliknutí na tlačítko přihlášení uživatelem aplikace otevře webový prohlížeč se stránkou Azure AD B2C aplikace, veškerou činnost uživatel vykonává na této webové stránce. Po dokončení webová stránka vrátí aplikaci JSON token, který obsahuje veškeré informace o úspěchu či neúspěchu přihlášení a uživateli. Tento token aplikace rozparsuje a adekvátně na výsledek zareaguje.

OpenWeatherMap

Aby bylo možné získat aktuální počasí na stanovišti, využívá aplikace dostupného aplikačního rozhraní pro získávání aktuálních dat o počasí Open Weather Map API[3]. Služba je tvořena třídou `OpenWeatherMapRestService.cs` pro volání Open Weather Map REST API a zpracování odpovědi. Další třídou, která je součástí této služby, je `WeatherData.cs`, tvořící strukturu pro ukládání přijatých dat o počasí (tato třída by se řadila do části model v rámci MVVM 5.1.2).

[2] https://www.nuget.org/packages/Microsoft.Azure.Mobile.Client/
[3] https://openweathermap.org/api

Získávání počasí funguje na základě města, které uživatel zadal jako to, na kterém se nachází stanoviště. Proto je nezbytné, aby uživatelem zadané město bylo bez gramatických chyb, i když jej může zadat v jakémkoliv jazyce.

Singleton

U všech výše zmíněných služeb je zapotřebí, aby existovaly pouze v jedné instanci v celé aplikaci a byly globálně přístupné. Všechny zmíněné služby jsou tak implementovány jako singleton. Singleton je návrhový vzor používaný při programování právě k tomu, když je potřeba, aby v celém programu existovala právě jedna instance určité třídy a také aby byl poskytnut globální přístupový bod k této třídě.

Singleton je v jazyce C# 4.4.2 možné implementovat různými způsoby. Obecně se však děje tak, že třída poskytuje přístup k jedné statické veřejné proměnné, která si drží instanci této třídy a konstruktor této třídy je pouze privátní, aby její instanci nemohl vytvořit nikdo jiný. Kromě dependency injection je to nejpoužívanější návrhový vzor v aplikaci.

Model-View-ViewModel

Všechny služby byly již popsané, a tak se nyní můžeme vrátit k základnímu kameni celé aplikace, a ten je tvořen architektonickým vzorem MVVM. Model-View-ViewModel je softwareový architektonický vzor, který umožňuje oddělení vývoje grafického uživatelského rozhraní (View) v jazyce XAML 4.4.3 od vývoje logiky (ViewModel) a datových objektů (Model) tak, že grafické uživatelské rozhraní není nijak závislé na žádné konkrétní platformě modelu.

ViewModel je určitým způsobem převodníkem hodnot, což znamená, že je odpovědný za převod datových objektů z modelu takovým způsobem, že objekty jsou jednodušeji zpracovávány a zobrazovány. V tomto ohledu je ViewModel spíše Model než View a zpracovává většinu, ne-li veškerou zobrazovací logiku. MVVM prvky je možné jednoduše předvést na příkladech z aplikace:

- **Model** — tvoří třídy reprezentující datové objekty. Příklad: `Apiary,cs`, třída tvořena veřejnými proměnnými reprezentujícími vlastnosti stanoviště.
- **View** — uživatelské rozhraní aplikace s veškerými grafickými prvky, to, co uživatel vidí. Příklad: `ApiaryDetailPage.xaml` společne s `ApiaryDetailPage.cs`, definují uživatelské rozhraní pro zobrazení a úpravu stanoviště.
- **ViewModel** — uchovává stav aplikace a zabezpečuje veškerou uživatelskou logiku. Udržuje si instance datových objektů z Modelu a utváří je do takové podoby, aby mohly být reprezentovány ve View. Příklad: `ApiaryDetailViewModel.cs` uchovává stav obrazovky `ApiaryDetailPage` ve vlastnostech, které například říkají, jestli má být zrovna zobrazen stav načítání nebo zdali je možné upravovat stanoviště. Také si udržuje instanci stanoviště, které je aktuálně zobrazováno, a všech úlů tohoto stanoviště, které jsou na stránce rovněž zobrazeny. V neposlední řadě obsahuje metody, navazující na jednotlivé interaktivní prvky z View. Všechny tyto vlastnosti jsou provázány s View za pomoci bindingu.

Binding je důležitou součástí vzoru MVVM. Bínding je způsob navázaní dat, které si udržuje ViewModel na View. V XAML aplikacích poskytuje jednoduchý a snadný způsob pro zobrazení a interakci s daty. Správa dat může být díky tomu kompletně oddělená od toho, jak jsou data prezentována.

```
<Entry
    Text="{Binding Name}"
    Placeholder="{x:Static
        resources:AppResources.Name}"
    Style="{StaticResource InputBox}"/>
```

Výpis 5.2: Ukázka použití bindingu v jazyce XAML.

Data binding umožňuje tok dat mezi grafickými prvky a datovými objekty v uživatelském rozhraní. Pokud je na některém grafickém prvku nastaven binding některé vlastnosti a tato vlastnost se ve ViewModel třídě změní, změna se ihned reflektuje na grafickém prvku. Tento vztah platí i obráceně, a pokud se změní vlastnost grafického prvku například uživatelskou interakcí, změna se projeví také ve ViewModel třídě. Takový vztah platí však jen pro tzv. two-way binding, data binding může být totiž dvojího druhu:

- **One-way binding** — data jsou provázaná pouze ze zdroje (ViewModel) k cíli (View). Uživatel tak nemůže modifikovat stav ve ViewModelu.
- **Two-way binding** — uživatel může modifikovat data skrze uživatelské rozhraní (View), a tak data aktualizovat u zdroje (ViewModel).

Na ukázce 5.2 můžeme vidět jednoduchý příklad použití bindingu na grafickém prvku `Entry`. Tento grafický prvek slouží k získání vstupních dat od uživatele, v tomto případě textu. Na vlastnosti `Text` je nastaven binding na proměnnou z ViewModelu `Name`, pokud tak uživatel zadá jakoukoliv hodnotu do tohoto pole, hodnota proměnné `Name` se změní.

5.1.3 Navigace

V aplikaci se vyskytují dvě obrazovky, které můžou být nastaveny jako hlavní obrazovky aplikace. Jedna pro přihlašování a druhá jako hlavní obrazovka po přihlášení. Obrazovka pro přihlášení je jednoduchá a je tvořena jednou `ContentPage` obrazovkou. U hlavní obrazovky po přihlášení se již vyskytují složitější prvky navigace.

Hlavní obrazovku po přihlášení tvoří `Master-Detail Page`, ta umožňuje rozdělit obrazovku na dvě části, kde jednu tvoří typické levé boční menu (Master) a druhou tvoří samotný obsah (Detail) [17]. Detail je v aplikaci tvořen instancí `NavigationPage`, díky které je možné v Detailu měnit různé obsahy/obrazovky.

Při použití architektonického vzoru MVVM je navigace mezi jednotlivými stránkami trochu složitější. Aby byl tento vzor totiž dodržen, View třídy by neměly obsahovat žádnou logiku, jenomže aby bylo možné přejít na jinou obrazovku, je potřeba použít proměnnou `Navigation` View třídy. Tato proměnná však nejde nijak navázat na ViewModel za pomocí bindingu. Vývojář má tak jen tři možnosti:

- Porušit vzor MVVM a pracovat s `Navigation` ve View třídách.
- Použít některou z dostupných knihoven, která navigaci v rámci MVVM usnadňuje, například PRISM[4].
- Použít vlastní řešení, jak s `Navigation` pracovat i ve ViewModel třídách.

[4] https://prismlibrary.com/

V aplikaci je použita třetí možnost. Vývojář si u ní musí dát pozor, aby byl dodržen stejný přístup k navigaci v celé aplikaci, jinak by v práci s navigací vznikal zmatek.

Možnost, jak přistupovat k instanci `Navigation` ve ViewModel třídách, je v práci zajištěna za pomoci dependency injection. Při vytváření instancí ViewModel tříd, jim View předá jako parametr instanci `Navigation`. Díky tomu je možné v každé ViewModel třídě přepínat v navigaci mezi jakýmikoliv obrazovkami.

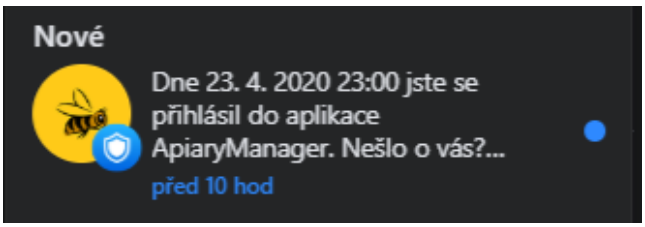
Na mockupu 3.6 z návrhu je vidět původní návrh, jak by navigace mezi jednotlivými obrazovkami měla probíhat. Oproti návrhu však několik obrazovek v aplikaci přibylo a navigace se také mírně změnila, ale v zásadě s návrhem sedí. Každá obrazovka kromě hlavní navíc obsahuje v horní části ikonu včely, která slouží jako návratový bod k domovské obrazovce.

5.1.4 Přihlašování

Pro příjemnou zkušenost s aplikací je v dnešní době nezbytné poskytnout uživatelům přihlášení za pomoci účtu na jejich oblíbené platformě. Uživatelům tak odpadá potřeba vytváření nového účtu a zapamatování si hesla pro přístup k aplikaci. V aplikace existují tedy dvě možnosti přihlášení, buď vytvořením nového účtu přímo v aplikaci nebo přihlášení za pomocí již existujícího účtu na Facebooku.

Přihlášení za pomocí Facebooku

Přihlašování za pomoci Facebooku má dvě roviny, jednou je autentizace uživatele a druhou je vyžádání přístupu k uživatelským údajům. Od uživatelů je tak možné vyžádat při přihlášení jakékoliv osobní informace, pro účely této práce to však není potřeba a od uživatelů jsou vyžadovány jen základní informace, jako jméno a email.

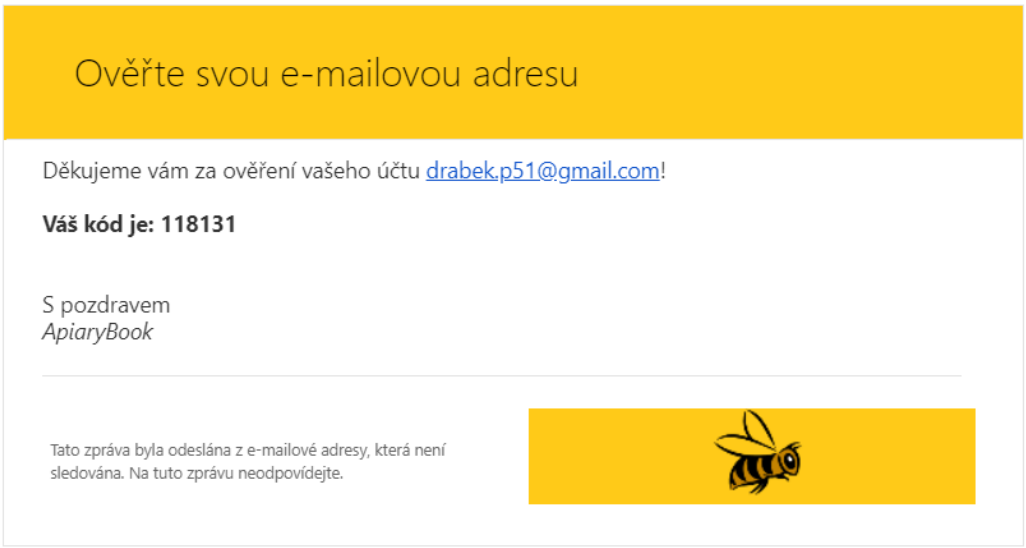


Obrázek 5.5: Upozornění o přihlášení do aplikace Deník včelaře na Facebooku.

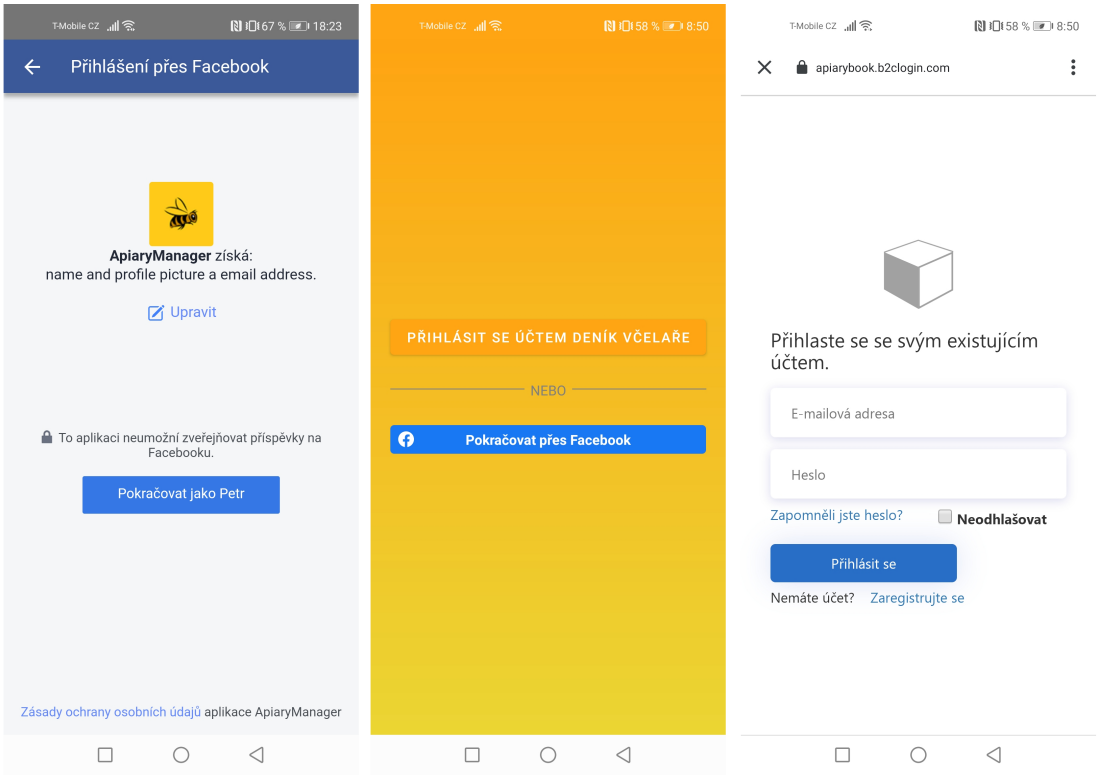
Při přihlašování přes Facebook se uživateli otevře webová stránka, kde se uživatel autentizuje svým účtem, po úspěšném přihlášení je vrácen autentizační token, díky kterému je možné uživatele autentizovat pro přístup do databáze. Uživateli přijde také upozornění 5.5 o přihlášení na Facebook.

Přihlášení za pomocí Azure AD B2C

Druhou možností přihlašování je za pomoci Azure AD B2C. Zajištění práce s tímto typem přihlašování zajišťuje služba `B2C Authentication` 5.1.2. Umožňuje uživateli registraci nového účtu, přihlášení, ale také obnovení zapomenutého hesla. Při přihlašování je po uživateli vyžadováno potvrzení emailové adresy 5.6, stejně tak jako při zapomenutém heslu, kdy uživatel tak potvrdí, že je opravdu majitelem tohoto účtu.



Obrázek 5.6: Potvrzení emailové adresy pro registraci do aplikace Deník včelaře.



Obrázek 5.7: Obrazovky pro přístup k přihlášení a správě účtu. Zleva: přihlášení za pomocí Facebooku, hlavní přihlašovací stránka aplikace, přihlášení za pomoci Azure AD B2C.

5.1.5 Asynchronní operace

Pokud pracujeme s uživatelským rozhraním a po kliknutí na tlačítko používáme metodu, jejíž běh trvá dlouho (typicky síťové operace nebo čtení velkého souboru), musí celá apli-

kace počkat na dokončení průběhu této metody. Jinými slovy pokud je jakýkoliv proces zablokován v synchronní aplikaci, aplikace se zasekne a přestane odpovídat, dokud není tento proces dokončen.

V takovém případě je řešením asynchronní programování. S použitím asynchronního programováním může aplikace obstarávat zároveň několik operací za pomoci využití více vláken. V .NET aplikaci můžeme využít všech výhod asynchronního programování s minimálním úsilí za pomoci využití klíčových slov `Async` a `Await` [16].

Jelikož aplikace pracuje s velkým množstvím síťových operací, async a await je využito v hojném zastoupení. Asynchronní metody jsou všechny, které pracují s databází. Získávání, přidávání, úprava a mazání jakýchkoliv položek se provádí asynchronně. Uživatel tak může například upravit položku v aplikaci a ihned dál pokračovat v používání aplikace, zatímco na pozadí se provede zapsání úpravy do databáze jak lokální, tak vzdálené.

```
private async Task UpdateApiaryExecuted()
{
    IsBusy = true;
    IsEditable = false;
    await ApiaryManager.Instance.UpdateApiaryAsync(Apiary);
    IsBusy = false;
}
```

Výpis 5.3: Neblokující metoda v jazyce C# pro upravení stanoviště.

Na příkladu 5.3 můžeme vidět ukázku použití asynchronní metody pro upravení stanoviště. Tato metoda je v deklaraci označena klíčovým slovem `async`, tím změníme způsob jejího překladu do stavového automatu, který bude zajišťovat spouštění částí kódu za a mezi jednotlivými await jako callback jednotlivých asynchronních volání. Po spuštění se tedy nejdříve provede přiřazení hodnot do proměnných `IsBusy` a `IsEditable`. Poté se pod klíčovým slovem `await` zavolá metoda `UpdateApiaryAsync`, jejíž běh trvá nějakou chvíli. Kód nacházející se za tímto voláním (přiřazení do proměnné `IsBusy`) se naplánuje jako callback a je opět spuštěn až poté, co skončí průběh metody `UpdateApiaryAsync`.

Optimalizace

Při několikanásobném volání operací s delší dobou běhu ve `foreach` cyklech (ale i dalších) nastává problém s dlouhým trváním provedení všech iterací v cyklu. Jako názorná ukázka může sloužit získání informací o počasí na všech stanovištích.

```
foreach (var apiary in Apiaries)
{
    apiary.Weather = await
        OpenWeatherMapRestService.Instance.GetWeatherData(apiary.Address);
    var index = Apiaries.IndexOf(apiary);
    if (index != -1)
        Apiaries[index] = apiary;
}
```

Výpis 5.4: Původní implementace získávání počasí pro každé stanoviště.

V původní implementaci 5.4 je v cyklu za pomocí klíčového slova `await` volána metoda pro získání informací o určitém stanovišti, která trvá delší dobu, jelikož se musí přes síť

nejprve dotázat na počasí a poté počkat na odpověď. V takové implementaci se při každé iteraci zavolá metoda pro získání počasí, následující kód se provede až po dokončení tohoto volání a teprve poté se může přejít k další iteraci pro získání počasí na další stanoviště.

Provedení všech iterací tohoto cyklu v takovém případě může trvat opravdu dlouho, hlavně v případě, kdy má uživatel více stanovišť a na informace o počasí na všech stanovištích musí čekat delší dobu.

Tento problém lze vyřešit jednoduše vytvořením kolekce obsahující metodu 5.5 zahrnující asynchronní operaci pro získání počasí na určitých stanovištích a poté zavolat klíčové slovo `WhenAll`, které zajistí neblokující čekání na dokončení všech těchto operací.

```
public async Task SetWeatherAsync(Apiary apiary)
{
    apiary.Weather = await
        OpenWeatherMapRestService.Instance.GetWeatherData(apiary.Address);
    var index = Apiaries.IndexOf(apiary);
    if (index != -1)
        Apiaries[index] = apiary;
}
```

Výpis 5.5: Neblokující metoda získávající počasí na stanovišti a vracející typ `Task` neboli asynchronní operaci.

```
var listOfTasks = new List<Task>();

foreach (var apiary in _apiaries)
{
    listOfTasks.Add(SetWeatherAsync(apiary));
}

await Task.WhenAll(listOfTasks);
```

Výpis 5.6: Optimalizovaná implementace získávání počasí na stanovištích.

Získávání počasí tak neprobíhá postupně jedno po druhém, ale zároveň. Dojde tak k velké úspoře času. I v případě jen pěti stanovišť trvá původní implementace v průměru 433 ms a optimalizovaná 5.6 jen 121 ms.

5.1.6 Použité NuGet balíčky

Nezbytným nástrojem moderního vývoje je mechanismus, pomocí kterého mohou vývojáři vytvářet, sdílet a používat užitečný kód. Takový kód je často zabalen do „balíčků“, které obsahují kompilovaný kód (DLL knihovny), spolu s dalším obsahem potřebným v projektech, které tyto balíčky používají. Pro .NET je **NuGet** tímto mechanismem pro sdílení kódu. Definuje, jak jsou balíčky pro .NET vytvářeny, hostovány, používány a poskytuje všechny nástroje k tomu potřebné.

NuGet balíček je v podstatě komprimovaná složka (ZIP) s příponou `.nupkg`, která obsahuje kompilovaný kód, další soubory spojeny s tímto kódem a soubor (manifest) popisující informace, jako je verze daného balíčku a další. Vývojář, který se chce podělit s užitečným kódem, může jednoduše vytvořit tento balíček a pro sdílení s ostatními vývojáři jej nahrát

na oficiální stránku pro sdílení NuGet balíčků[5]. Vývojář, který tento kód chce poté použít, jej jednoduše za pomocí NuGet Package Manageru přidá do svého projektu a okamžitě jej může používat. NuGet se sám poté postará o veškeré podrobnosti, nezbytné k zajištění funkcionality celého mechanismu sdílení.

V mobilní aplikace je těchto balíčků použito mnoho, ať již přímo, nebo v závislosti na další balíčky. V následujícím seznamu jsou popsány jen ty nejpodstatnější z nich.

Microsoft.Azure.Mobile.Client Tato knihovna poskytuje veškeré potřebné funkce k propojení mobilní aplikace s AzureApp Service a aplikace hostované v ní. Díky této knihovně tak můžeme pracovat s datyz databáze, autentizovat uživatele, ale také pracovat s offline synchronizací dat a používatpush notifikace.

Microsoft.Azure.Mobile.Client.SQLiteStore Je dalším nezbytným balíčkem pro zajištění offline synchronizace dat. Konkrétně umožňuje ukládat data lokálně přímo na zařízení v SQLite databázi.

Microsoft.Identity.Client Tento balíček obsahuje binární soubory Microsoft Authentication knihovny pro .NET. Umožňuje v aplikaci používat Azure AD B2C autentizaci uživatelů a zpracovávat příchozí JSON tokeny.

Newtonsoft.Json Velmi populární framework, pro zpracování a práci s JSON soubory. V aplikaci je využit hlavně pro zpracování všech příchozích JSON tokenů.

Plugin.FacebookClient Balíček, který umožňuje použit přihlašování za pomocí Facebooku v aplikaci, vyžadovat povolení k osobním údajům uživatelů a zpracovávat odpovědi.

Rg.Plugins.Popup Plugin pro Xamarin.Forms umožňující otevírat jakoukoliv stránku jako popup. V aplikaci je použit k vytvoření vlastních vzhledů, funkčnosti popupů a zajištění navigace pro zobrazování a skrytí těchto popupů.

Syncfusion.Xamarin.SfChart Poskytuje použití interaktivních a moderních grafů v aplikaci.

Xamarin.Forms.PancakeView Velmi používaný balíček, který umožňuje graficky přizpůsobovat `ContentView` dle libosti. V aplikaci je využit hlavně na definování barevných přechodů a různě zaoblených rohů.

5.2 Prototyp úlové váhy

Pro vytvoření fungujícího prototypu úlové váhy bylo potřeba vytvořit nejen program, který generuje váhu úlu a posílí data do databáze, ale také obstarat a sestavit požadovaný hardware. Následující sekce je tak rozdělena do dvou části popisující řešení těchto problémů.

5.2.1 Software

Pro vývoj programu simulující vážení úlu bylo použito vývojové prostředí Arduino IDE[6]. Toto prostředí umožňuje rychlý vývoj, nahrávání programů na vývojové desky, ale také sledování konzolových výpisů na daném portu s vývojovou deskou komunikujícím.

[5] https://www.nuget.org/
[6] https://www.arduino.cc/en/Main/Software

Výsledný program je napsán v jazyce C a funguje na jednoduchém principu uspávání a probouzení zařízení v přesně daných časových intervalech pro odeslání dat na server. V každém probuzení se zařízení nejdříve přípojí k WiFi síti (název a heslo k síti musí být součástí programu), poté získá aktuální čas z NTP serveru[7], odešle údaje o váze úlu, nastaví časovač, který zařízení probudí za dalších 30 minut a uspí zařízení do hlubokého spánku. Zařízení v hlubokém spánku nespotřebovává téměř žádnou energii. Energetická náročnost tohoto prototypu je tak minimální a i na baterii dokáže běžet opravdu dlouho, v řádech měsíců.

Odeslání údajů o váze úlu, jak již z předchozího popisu vyplývá, probíhá za pomocí Wi-Fi sítě. Údaje o váze je tak možné odesílat v podobě JSON souboru přímo na REST API ASP.NET aplikace. Pro odeslání je tak nutné pouze sestavit čas ve správném formátu, váhu úlu a GUID úlu, kterého se toto vážení týká, do textového řetězce reprezentujícího JSON soubor. Ten je pak odeslán za pomocí knihovny `HTTPClient` v HTTP POST metodě přímo na REST API. ASP.NET aplikace poté tato příchozí data zpracuje a uloží do databáze, odkud mohou být ihned promítnuty do mobilní aplikace.

5.2.2 Hardware

Pro sestavení úlové váhy byly použity komponenty tak, jak byly představeny v návrhu 3.5. Pro propojení jednotlivých komponent bylo použito nepájivé pole, aby se dalo s komponenty v průběhu sestavování lehce manipulovat a testovat jejich funkčnost.

Na obrázku 5.8 lze vidět celou sestavu prototypu úlové váhy. Její hlavní část tvoří vývojová deska ESP32, kterou je vidět v dolní části obrázku s rozsvícenou červenou diodou. Ostatní části prototypu tvoří napájení této vývojové desky. Napájení je tvořeno baterií z dvou článků s jmenovitým napětím 3.7 V a celkovou kapacitou 5000 mAh. Baterie poskytují energii v případě, kdy solární panel není v provozu, ale pokud solární panel v provozu je, přebytečná energie je do této baterie ukládána.

Pro sestavení funkční úlové váhy by stačilo již jen k vývojové desce připojit tenzometry umístěny pod úlem a část kódu, která by převáděla hodnoty z tenzometrů na reálnou váhu. Všechny komponenty kromě solárního panelu poté umístit do voděodolného obalu a z prototypu by tak mohla být funkční úlová váha.

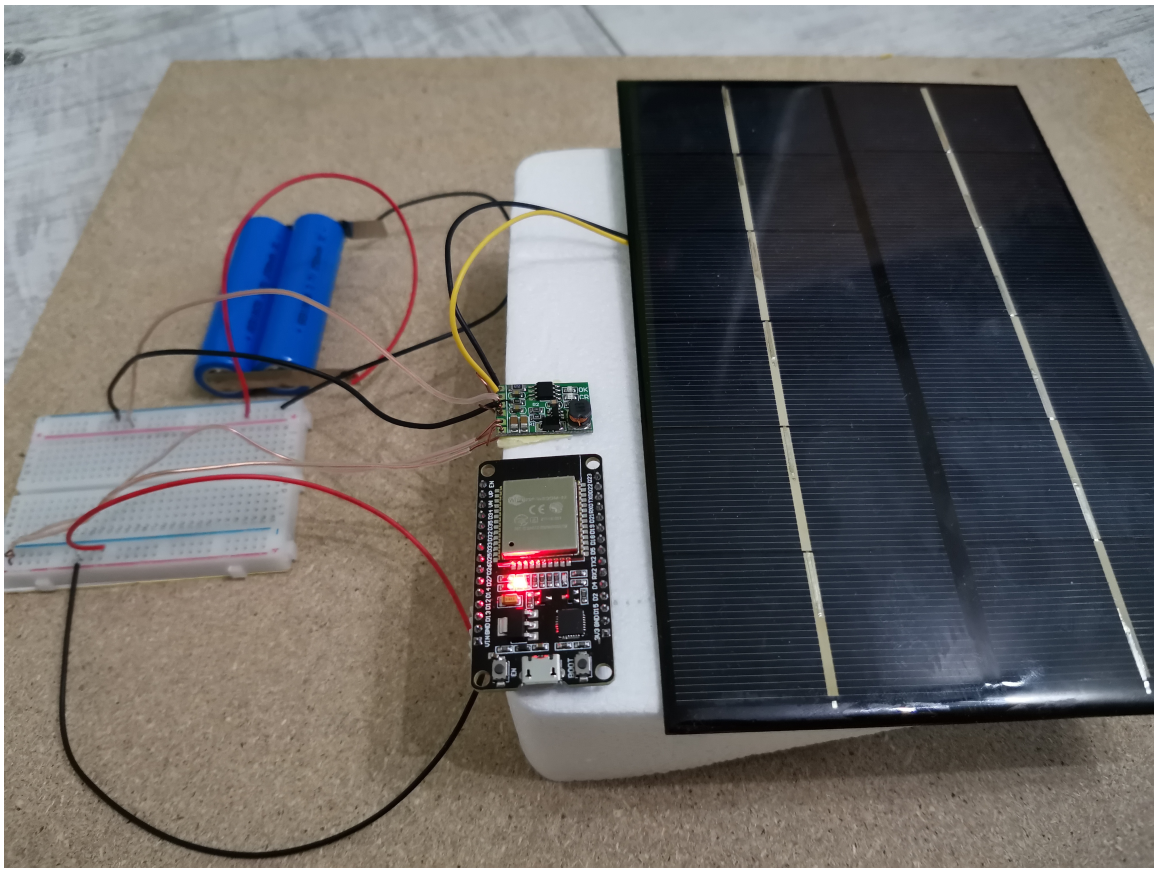
5.3 Serverová část

Serverovou část reprezentuje databáze a ASP.NET aplikace, která je nezbytná pro fungování celého systému. ASP.NET aplikace je implementována podle vzoru[8] určeného k vytváření těchto aplikací – sloužících pro propojení databáze s mobilní aplikací – na Azure.

Implementace obsahuje deklaraci všech datových objektů z návrhu 3.2 ve třídách, tyto třídy jsou za pomocí Entity Frameworku namapovány přímo do tabulek v databázi. Oproti návrhu chybí tabulka `uživatel`, která díky použití Azure AD B2C tak v práci není potřeba. V datových objektech krmení a léčení zase přibyla položka pro přidání poznámky k těmto událostem. Pro fungování REST API pro přístup k těmto datovým objektům v databázi obsahuje aplikace také třídy implementující metody `GetAllItems`, `GetItem`, `PatchItem`, `PostItem` a `DeleteItem`, kde `Item` reprezentuje název datového objektu, pro práci s databází.

[7] https://www.ntppool.org/

[8] https://github.com/Azure/azure-mobile-apps-quickstarts/tree/master/backend/dotnet/Quickstart



Obrázek 5.8: Sestavený prototyp úlové váhy.

Kapitola 6

Testování mobilní aplikace

Jednou z hlavních součástí ve vývoji mobilních aplikací je jejich testování. Testováním lze odhalit chyby, ale také zjistit místa, která kupříkladu nejsou vhodně navržená a implementovaná k reálnému použití. Testování mobilní aplikace Deník včelaře je provedeno z několika důvodů:

- Odhalit funkcionální, ale i grafické vady aplikace.
- Otestovat uživatelskou přívětivost a intuitivnost aplikace.
- Zjistit jakékoliv možnosti k vylepšení aplikace.
- Ověřit použitelnost mobilní aplikace v praxi.

Následující sekce obsahuje návrh testů, popis jejich provedení a vyhodnocení výsledků na dvou odlišných způsobech testování na uživatelích. Prvním z nich je praktické testování přímo na včelnici 6.1 s včelařem, který bude používat tuto mobilní aplikaci poprvé a v další části bude představeno vyhodnocení dlouhodobějšího testování mobilní aplikace 6.2 včelaři z místního spolku.

6.1 Praktické testování

Praktické testování je uskutečněno k ověření intuitivnosti aplikace a použitelnosti v praxi. Cílem tohoto testování je zjistit, jak uživatelé budou schopni mobilní aplikaci používat, zdali budou schopni intuitivně vykonat veškeré úkony z případů užití, rychlost provedení těchto úkonů a míst, kde by aplikace mohla být více pochopitelná uživatelům.

Návrh konkrétních testů je uspořádán do tabulky 6.1, kde ke každému případu užití je nejdříve rozepsán způsob, jak tento test provést, na jaká místa se zaměřit, ale také jak tento test vyhodnotit. Jednotlivé testy budou provedeny s uživatelem přímo na včelnici, aby byla ověřena použitelnost mobilní aplikace co nejlépe v praxi. Jelikož nechci příliš zatěžovat včelaře, s kterým bude testování provedeno, a navíc k ověření použitelnosti není potřeba testovat všechny části aplikace, navržené testy budou zaměřeny na hlavní a nejčastější případy užití tak, aby byla ověřena intuitivnost ovládání aplikace a rychlost zaznamenávání jednotlivých položek. Testy budou uspořádány postupně, aby co nejlépe odpovídaly reálnému používání aplikace. Uživatel tak nejprve vytvoří nový účet v aplikaci, poté mu bude ponechán krátký čas na seznámení s aplikací a přejde se k testům zaznamenávání konkrétních položek a zobrazení vyfiltrovaných statistik.

Tabulka 6.1: Návrh testů pro testování s uživatelem v praxi.

Případ užití	**Způsob testování**	**Způsob vyhodnocení**
Vytvoření nového účtu za pomoci emailové adresy.	Předat mobilní aplikaci uživateli, který ji dříve nepoužíval, a zadat mu úkol – vytvořit nový účet.	Bylo uživateli jasné, jak v aplikaci vytvořit nový účet? Jak dlouho celý proces vytvoření účtu trval? Podařilo se správně zaslat ověřovací kód na emailovou adresu?
Vytvoření několika nových stanovišť s danými vlastnostmi.	Předat uživateli informace ke třem stanovištím a nechat jej tyto stanoviště přidat do aplikace. Zároveň uživatele požádat, aby při vytváření stanoviště hlasitě přemýšlel a jeho myšlenkové pochody popisoval.	Přišel uživatel ihned na to, jak stanoviště/úl přidat? Poslouchat a pozorovat uživatele, která část přidání stanoviště/úlu mu zabere nejvíce času. Na tuto část se zaměřit a promyslet možnosti zlepšení.
Vytvoření několika úlů s danými vlastnostmi.	Způsob testování je stejný jako v předchozím testu, až na to, že nyní bude uživatel vytvářet úly.	
Zaznamenání prohlídky.	Nechat uživatele provést prohlídku včelstva na referenční včelnici a po provedení všechny informace zaznamenat do mobilní aplikace ke konkrétnímu úlu.	Opět se zaměřit na pozorování kritických míst, která uživatele při zadávání brzdí. Vyplnil uživatel všechny položky prohlídky? Nechyběla uživateli nějaká položka, aby mohl zadat, co chtěl?
Vytvoření hromadného medobraní.	Představit uživateli scénář, kdy probíhá medobraní celého stanoviště a nechat jej zaznamenat toto medobraní. Uživatel nezná počet vytočeného medu z jednotlivých včelstev, ale jen sumu z celého stanoviště.	Cílem toho testu je odhalit, zdali uživatel bude schopen zadat hromadné medobraní na celém stanovišti. Pokud uživatel nepochopí přidávání hromadného medobraní a krmení, promyslet s uživatelem možná zlepšení.
Zobrazení konkrétních statistik.	Předat uživateli mobilní aplikaci s účtem, na kterém se nachází referenční data, a zadat mu úkol – zobrazit statistiky, kolik bylo vytočeno medu na včelnici XY v roce 2020.	Dokázal uživatel intuitivně přejít do sekce statistik? Pozorovat, jakým způsobem filtruje data, je mu požití filtrů pochopitelné? Chybí uživateli ve statistikách nějaká položka?

K uskutečnění praktického testování jsem požádal známého včelaře (dále jen uživatel), zda by tyto testy se mnou neabsolvoval. Ten mé žádosti vyhověl a testy byly prováděny na jeho včelnici v obci Strání jako součást běžné prohlídky včelstev. Uživatel neměl předchozí zkušenosti s podobnou aplikací a s testovanou aplikací se seznámil poprvé při testech. Samotné testování proběhlo bez komplikací, průběh a vyhodnocení jednotlivých testů je představen v následující části.

Vytvoření nového účtu za pomoci emailové adresy.

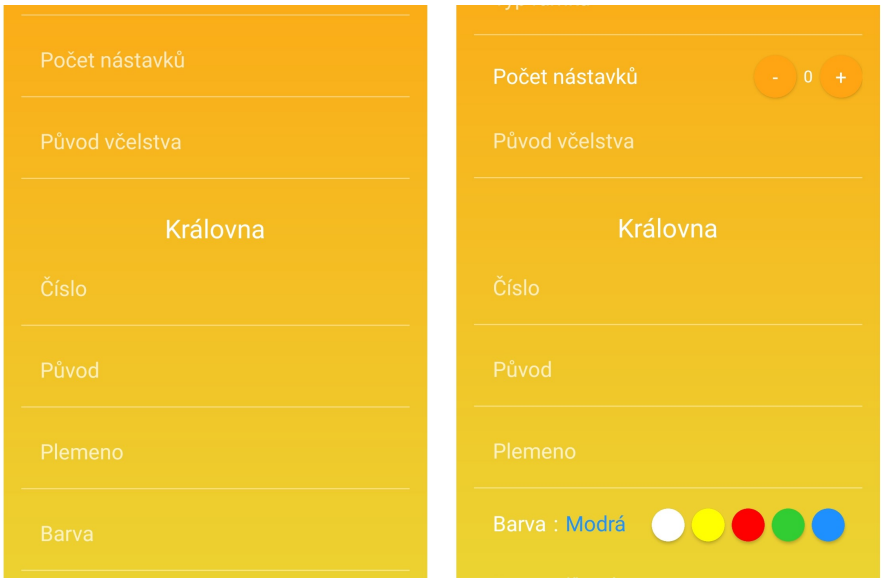
První test proběhl skvěle, uživatel byl schopen všechny kroky sám splnit, email s ověřovacím kódem přišel bez prodlení a uživatel měl do pár minut vytvořen nový účet.

Vytvoření několika nových stanovišť

Druhým testem bylo vytvoření tří stanovišť, tento test také netrval dlouho a uživatel měl stanoviště vytvořená. Uživatel si správně spojil slovo "přidat" s tlačítkem na hlavní obrazovce a dále jej aplikace navedla. V průběhu nebyl zaznamenán žádný bod, který by uživatele brzdil.

Vytvoření několika úlů

Uživatel v přidávání postupoval ze začátku stejně jako s přidáváním stanovišť. Přidání úlů má však již více informací k vyplnění a přidávání trvalo delší dobu. Společně jsme se zamysleli nad možným zlepšením, sám uživatel navrhl možnost výběru matky z pěti používaných barev, na testované verzi se barva zadávala textem. Dalším urychlením přidávání úlů se stala možnost odklikat počet nástavků, místo zadáváním počtu na klávesnici. Rozdíly mezi testovanou a konečnou verzí lze vidět na obrázku 6.1. Ve výsledné implementaci se navíc barva královny automaticky určuje podle aktuálního roku.



Obrázek 6.1: Rozdíl mezí původní implementací přidávání úlů a implementací po testování.

Zaznamenání prohlídky

Uživatel provedl běžnou prohlídku na jednom svém včelstvu a poté mu byl zadán úkol k zaznamenání této prohlídky do mobilní aplikace. Přidání prohlídky proběhlo rychle a nebyl nalezen žádný způsob, jak přidávání prohlídek ještě urychlit. Uživatel však nevyužil možnost zaznamenat u prohlídky mezerovitost plodu (také si všiml hrubky v aplikaci ve slově mezerovitost, které původně bylo napsáno jako "mezerivost"). Pro testovaného uživatele není tento údaj podstatný. Podle průzkumu mezi uživateli 2.3 si tento údaj zaznamenává zhruba 25 % respondentů, není to mnoho, ale pořád se jedná o podstatnou část včelařů. Prozatím bylo zaznamenávání mezerovitosti plodu u prohlídky v aplikaci ponecháno.

Vytvoření hromadného medobraní

Podstatou hromadného medobraní a krmení v aplikaci je zaznamenání této činnosti na všech úlech daného stanoviště. Pokud tak uživatel nesleduje medný výnos konkrétních úlů, ale odebírá med na celém stanovišti a sleduje výnos celého stanoviště, využije tuto možnost. V aplikaci tak jen vybere stanoviště a zadá množství pro celé stanoviště, to se pak rozpočítá na jednotlivé úly.

Uživateli byl zadán úkol zaznamenat medobraní celého stanoviště. Hromadné medobraní bylo uživatelem pochopeno správně a v zaznamenávání nenastal žádný problém.

Zobrazení statistik s použitím filtru

Uživatel nejdříve hledal statistiky v levém bočním menu, brzy však pochopil význam ikony statistiky na hlavní obrazovce a do této sekce přešel. S vyfiltrováním statistik podle roku a konkrétního stanoviště již žádný problém nenastal.

V důsledku tohoto testu byla do levého bočního menu přidány všechna tlačítka z hlavní obrazovky v textové podobě.

6.2 Dlouhodobé testování

K odhalení funkcionálních a grafických vad aplikace, zjištění jakékoliv možnosti k vylepšení aplikace a ověření použitelnosti v praxi byla mobilní práce předána k testování čtyřem včelařům z místní organizace. Tito včelaři používali mobilní aplikaci po dobu dvou týdnů a poté jsem s nimi vedl společný rozhovor o spokojenosti s aplikací, chybějících ale i přebývajících částech aplikace a chybách pokud na některé narazili.

Testovaní včelaři popisovali zkušenost s mobilní aplikací pozitivně a s nadšením, nicméně některé nedostatky nalezli a přišli také s několika možnými nápady na vylepšení. V následující sekci jsou popsány nejprve nalezené chyby, jejich řešení a poté navržené nápady na zlepšení.

6.2.1 Nalezené chyby

V následující sekci jsou popsány chyby a nedostatky na které testující včelaři v průběhu testování přišli a ve společném rozhovoru zmínili.

Zaseknutí aplikace po zrušení přihlášení

Jeden z včelařů zaznamenal stav aplikace, kdy při použití přihlašování za pomoci Azure AD B2C a zrušení tohoto přihlášení, se aplikace zasekla a zobrazovala jen točící se kolečko. Tento

stav byl důsledkem špatně napsané metody `LoginExecute` ve třídě `LoginViewModel.cs`, kdy po zrušení stránky s přihlášením nedošlo k nastavení hodnoty `false` do proměnné `IsBusy`, která indikuje průběh načítání. Tato chyba byla úspěšně opravena.

Odhlášení z Facebook účtu

Pokud se uživatel odhlásil z mobilní aplikace, kde byl přihlášen za pomoci svého Facebook účtu, na přihlašovací obrazovce zůstalo tlačítko pro Facebook přihlášení s textem `"Odhlásit se"` a uživatel se musel za pomoci tohoto tlačítka odhlásit ještě jednou, aby mohl použít rozdílný Facebook účet.

Tento problém se bohužel nepodařilo z aplikace odstranit. Tlačítko pro přihlášení za pomoci Facebook účtu má své specifické chování, které v tomto použití nelze nijak z vnějšku ovlivňovat. Možným řešením do budoucna by mohlo být umístění stejného tlačítko do levého menu hlavní obrazovky, to ale jen v případě, kdy je uživatel přihlášen za pomocí Facebooku.

Zaznamenání váhy medu s desetinným číslem

Jeden z testovaných včelařů přišel na chybu, že u medobraní nejdou zadat kilogramy s desetinným místem. Včelaři tak měli možnost zaznamenat jen celé kilogramy. Stejná chyba se nacházela i u přidávání krmení, kde včelaři měli možnost zadat jen celé kilogramy. Tato chyba byla v aplikaci odstraněna, příčina byla použití datového typu `integer` u těchto položek, řešením bylo použít datový typ `double`, který zaznamenávání čísel s desetinnými místy umožňuje.

6.2.2 Nápady na vylepšení

V rámci společného rozhovoru s testovanými včelaři byly probírány také možná vylepšení mobilní aplikace. Některá tato vylepšení jsou ve výsledné práci již zahrnuta, ale některá jsou naplánována až do dalšího vývoje, protože bylo složitější je do aplikace jednoduše zakomponovat.

Průměr na jeden úl ve statistikách

Jeden z včelařů podal návrh, že ve statistikách by kromě celkové množství podaného krmiva a vytočeného medu mohlo být množství na jeden úl. Mezi ostatními včelaři došlo ke shodě, že by tento údaj byl užitečný. Do statistik byl tedy tento údaj přidán přímo pod celkové množství.

Zaznamenání přidaných nástavků a rámků

K další shodě došlo u zaznamenávání si přidaných nástavků, stavebních rámků, souší a mezistěn v průběhu prohlídky. Taková změna však již vyžaduje migraci databáze, upravení ASP.NET a také mobilní aplikace. Toto zlepšení není součástí odevzdané práce, ale v budoucím vývoji bude do aplikace určitě zahrnuto.

Kapitola 7

Závěr

Cílem práce bylo vytvořit multiplatformní mobilní aplikaci nahrazující klasický papírový deník pro evidenci včelařské činnosti. Vytvořit prototyp úlové váhy jako demonstraci možného propojení váhy s mobilní aplikací. V neposlední řadě také implementovat fungující serverovou část pro práci s databází a umožnit propojení úlové váhy s mobilní aplikací.

Úvod práce se věnuje samotnému problému včelařského deníku. Je zde představena základní včelařská terminologie pro zasvěcení laiků do světa včel, analýza již existujících řešení nahrazujících včelařský deník, ale také průzkum mezi včelaři pro zjištění, jaká data by si rádi do včelařského deníku zaznamenávali.

Další část je zaměřena na návrh databáze, je zde představen datový návrh s vysvětlením jednotlivých položek v databázi a entitně-vztahový model databáze. Dále návrh samotné mobilní aplikace, zamyšlení se nad případy užití a návrh některých obrazovek za pomocí skic. V neposlední řadě také návrh úlové váhy, zvolení jednotlivých hardwarových komponent, ale také softwaru. Po návrhu následuje část s použitými technologiemi pro vývoj mobilní aplikace, serveru, ale také propojení s Azure službami. Úvod této části je věnován problému multiplatformního vývoje mobilních aplikací, dále představení vývoje v Xamarinu a Xamarin.Forms a nakonec technologie k serverové části a Azure.

Nedílnou součástí práce je implementace všech navržených částí. V kapitole zabývající se implementací je představeno grafické rozhraní části mobilní aplikace, ale hlavně představení a popis architektury celého systému a jeho komponent. Úspěšně se podařilo implementovat multijazyčnou mobilní aplikaci nahrazující včelařský deník, kterou navíc uživatelé mohou používat i bez přístupu k internetu (díky offline synchronizaci dat), který na včelnici často nemají. Krátkou ukázku z mobilní aplikace lze shlédnout na https://youtu.be/BwscEJfCZtA. Do aplikace lze přistupovat existujícím účtem na Facebooku nebo vytvořením nového účtu této aplikace, jeden účet pak může zároveň používat více uživatelů (například v případě více zaměstnanců včelařské farmy). Mobilní aplikace byla testována s několika včelaři a výsledky testů jsou v implementaci také zahrnuty. Tyto testy bohužel probíhaly pouze na Android zařízeních, jelikož se mi vlivem koronavirové krize nepodařilo dostat k Mac zařízení nezbytném k doladění, sestavení a otestování aplikace pro iOS. Aplikace se v průběhu testování setkala s pozitivním přijetím a nadšením od těchto včelařů.

Podařilo se také implementovat veškerou serverovou část nezbytnou k fungování aplikace a sestavit funkční prototyp úlové váhy, který odesílá vygenerované údaje o váze v pravidelných intervalech. Prototyp ke své činnosti navíc nepotřebuje externí napájení, ale je napájen za pomoci systému se solárním panelem a baterií.

Ve vývoji mobilní aplikace chci pokračovat i po odevzdání bakalářské práce. Možností dalšího vývoje jsou jednoduššího i složitějšího charakteru. Mezi ty jednoduché patří například možnost zaznamenání si počtu přidaných nástavků a rámků v průběhu prohlídky, umožnit uživatelům ukládat si do aplikace pořízené fotografie nebo identifikace úlů za pomocí QR kódů. Možnosti dalšího vývoje složitějšího charakteru jsou rozpoznávání počtu spadu kleštíka včelího z bílé podložky nebo analýza stavu včelstva za pomoci zvuku, který včelstvo vydává. Aplikaci chci také doladit pro iOS zařízení a publikovat v App Store a Google Play, aby mohla sloužit nejen českým, ale také zahraničním včelařům.

Literatura

[1] *App Service* [online]. Microsoft [cit. 2020-05-3]. Dostupné z: https://azure.microsoft.com/cs-cz/services/app-service/.

[2] *Azure Active Directory B2C* [online]. Microsoft [cit. 2020-05-3]. Dostupné z: https://azure.microsoft.com/en-us/services/active-directory/external-identities/b2c/.

[3] *Bees Declared To Be The Most Important Living Being On Earth* [online]. The Science Times [cit. 2020-01-17]. Dostupné z: https://www.sciencetimes.com/articles/23245/20190709/bees-are-the-most-important-living-being-on-earth.htm.

[4] *Flutter Technical overview* [online]. Google Inc. [cit. 2020-04-19]. Dostupné z: https://flutter.dev/docs/resources/technical-overview.

[5] *Mobile Operating System Market Share Worldwide* [online]. StatCounter [cit. 2019-04-22]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[6] *Native vs Hybrid vs Cross-platform Mobile app – What to Choose?* [online]. EGO CREATIVE INNOVATIONS [cit. 2020-02-13]. Dostupné z: https://www.ego-cms.com/post/native-vs-hybrid-vs-cross-platform-mobile-app-what-to-choose.

[7] *Nemoci včel se šíří i kvůli rozmachu laiků. Včelaření se stává symbolem ekologického smýšlení, říká biolog* [online]. Lidovky.cz [cit. 2020-01-20]. Dostupné z: https://www.lidovky.cz/domov/vcelam-pomaha-sad-i-nesekany-travnik-podle-biologa-se-vcelareni-se-stava-symbolem-ekologickeho-smysl.A190615_135302_ln_domov_ele.

[8] *React Native* [online]. Facebook Inc. [cit. 2020-04-19]. Dostupné z: https://reactnative.dev/.

[9] *A tour of the C language* [online]. Microsoft [cit. 2020-04-28]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.

[10] *Use Case Diagram* [online]. SmartDraw [cit. 2019-12-25]. Dostupné z: https://www.smartdraw.com/use-case-diagram/.

[11] *Včely byly prohlášeny nejdůležitějším organismem na planetě* [online]. Echo24 [cit. 2020-01-18]. Dostupné z: https://echo24.cz/a/Sytbp/vcely-byly-prohlaseny-za-nejdulezitejsi-organismus-na-planete-pritom-jsou-v-ohrozeni.

[12] *Xamarin mobile apps* [online]. Microsoft [cit. 2020-04-27]. Dostupné z: https://dotnet.microsoft.com/apps/xamarin/mobile-apps.

[13] *Xamarin.Forms Material Visual* [online]. Microsoft [cit. 2020-05-10]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/visual/material-visual.

[14] *XAML overview in WPF* [online]. Microsoft [cit. 2020-04-28]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/desktop-wpf/fundamentals/xaml.

[15] *ESP32 Technical Reference Manual* [online]. Espressif Systems, 2019 [cit. 2020-01-19]. Dostupné z: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

[16] Davies, A. *Async in C 5.0*. O'Reilly, 2012. ISBN 978-1449337162.

[17] Hermes, D. *Xamarin Mobile Application Development*. APress, 2015. ISBN 1484202155.

[18] kolektiv, V. V. a. *Včelařství*. Brázda, 2003. ISBN 80-209-0320-8.

[19] Lacko Luboslav. *Vývoj aplikací pro Android*. COMPUTER PRESS, 2015. ISBN 978-80-251-4347-6.

[20] Lacko Luboslav. *Vývoj aplikací pro iOS*. COMPUTER PRESS, 2018. ISBN 9788025149423.

[21] Margolis, M. *Arduino Cookbook*. O'Reilly Media, 2nd edition, 2011. ISBN 978-1449313876.

[22] Tew, J. E. *Nepostradatelný rádce včelaře*. REBO International CZ, 2015. ISBN 978-80-255-0905-0.

[23] Xamarin, I. *Creating Mobile Apps with Xamarin.Forms*. Microsoft Press, 2016. ISBN 978-1-5093-0297-0.